Background Subtraction for Urban Traffic Monitoring using Webcams

- Master Thesis –

By: Mark Smids {msmids@science.uva.nl}
Supervised by: Rein van den Boomgaard {rein@science.uva.nl}
University: Universiteit van Amsterdam, FNWI
Date: December 2006

Abstract
This research proposes a framework of a visual urban traffic monitoring system capable of obtaining the traffic flow at intersections of a city in a cheap way. The main focus of this research is on the background subtraction and shadow elimination component of this proposed system. Two different background subtraction techniques are implemented and compared: a deterministic approach using an adaptive background model and an statistical approach using a per-pixel Gaussian mixture model (GMM) to obtain information about the background. Furthermore, as a way of testing the two background subtraction algorithms, a smart surveillance system is built using a video summarization technique, which only stores those parts of the scene that include traffic. The system is evaluated using traffic scenes recorded under different weather conditions. Subtraction results of the deterministic subtraction approach show some important limitations. The total area of an foreground object is often not fully detected and objects in the scene that have become part of the background are wrongly classified as foreground objects for a long period of time. Results based on the GMM subtraction approach are very promising. The limitations mentioned above do not apply for this method and is therefore far more effective in the urban traffic setting.

Keywords: background subtraction, video summarization, shadow detection, Gaussian mixture model, OpenCV, smart surveillance

FACULTEIT DER NATUURWETENSCHAPPEN, WISKUNDE EN INFORMATICA

Table of Contents

1. Introduction 3

2. System Requirements and setting 4
2.1. The Setting 4
2.2. The tasks and requirements of a robust urban traffic monitor system 4

3. System overview 5

4. Literature on background subtraction algorithms 6
4.1. Pre-processing 6
4.2. Background Modelling 9
4.3. Foreground detection 11
4.4. Data Validation 12
4.5. Shadow detection and elimination 13

5. Implementations 15
5.1. Deterministic background model 15
5.2. Statistical background model (mixture of Gaussians) 19
5.3. Shadow detection and removal 23
5.3.1. Shadow detector for the deterministic background subtractor 23
5.3.2. Shadow detector for the statistical background subtractor 24
5.4. Video Summarization 25
5.5. How to use the software 26

6. Evaluation 27
6.1. Types of videos 30
6.2. Parameter tuning 30
6.3. Comparison measure 32
6.4. Comparison results on practical video's 33

7. Conclusions 37

8. Future work 38

9. References 40

Appendix **A**. Graphical derivation of the S_{RGB} formula 43

Appendix **B**. Rewritten formula for colour distortion 43

1. Introduction

Monitoring traffic using a vision oriented system in crowded urban areas is a challenging task. To have an overview of the traffic flow through the city can be very advantageous when new roads in the city are built or reconstructed. Also traffic light wait times can be programmed more efficiently when long-term traffic flow through the city is known. Current approaches of monitoring traffic includes manual counting of vehicles, or counting vehicles using magnetic loops on the road. The main drawback of these approaches, besides the fact that they are expensive, is that these systems only count. Vehicle classification, speed and location registration for individual vehicles and vehicle size, are properties that are not, or only limited, supported by these systems.

In this Master thesis a much cheaper and versatile traffic monitor system is proposed using a vision based approach. In this setting, traffic at intersections in cities are monitored using simple cameras (webcams) that are located on a high spot somewhere near these intersections. A typical video frame obtained from such cameras is displayed in figure 1. Most existing research and applications on traffic monitoring focus on monitoring cars on highways. As will become clear in chapter 2, successful monitoring traffic at crossroads in crowded urban areas is far more difficult. Using a vision based approach for monitoring, the mentioned drawbacks of classic systems no longer exist. Further more the flow of the traffic in the city can easily be displayed at a central place by combining information from each video stream.

Figure 1 *– A typical intersection in an urban area, where a webcam is placed at some high location*

Although a full framework of the proposed monitoring system will be given, this research focuses mainly on one component in the framework: the background subtraction and shadow elimination. An extensive overview of methods and algorithms concerning background subtraction and shadow elimination will be discussed in the literature section. Two different background subtraction algorithms suitable for our urban setting are actually implemented (a statistical and a deterministic approach) and compared. The algorithms are then finally evaluated by testing them on a number of videos of traffic flow at crossroads. To test the system in real-time and for a long period of time, monitoring using a webcam is also possible with the created software. The software can also function as a "smart surveillance" system where only the scenes of interest are recorded to disk using video summarization techniques. The video test samples, project information, this thesis in digital format, and the software itself can be obtained from a website [29].

This Master thesis will now continue in the following way: in chapter 2 the requirements of a robust urban traffic monitor system are discussed. Also the setting and properties of a typical crossroad in an urban area is given here. Chapter 3 gives an overview of the proposed system and defines its components. Existing literature and methods on background subtraction and shadow elimination will be discussed in chapter 4. Chapter 5 includes the description of the two implemented background subtraction algorithms and a

short user manual for the software. Finally in chapter 6 the two implemented approaches are evaluated and compared. The goal of this research is to give an answer to the question which approach, a deterministic or statistical approach, performs better background subtraction results in the given setting.

2. System Requirements and setting

2.1. The Setting

In this section the environmental setting of this research will be discussed and, given this setting, the requirements of a traffic monitor system will be given. Getting a complete overview of requirements and possible difficulties that might occur is an important first step in the design of such a complex system. The first assumption is that the focus is on the monitoring of *urban* traffic and monitoring the *flow* of traffic through a city. Typical places to mount camera's will logically be at important intersections and crossroads. By identifying vehicles and keep track of their direction in each stream, an overview of the flow of the traffic can be realized in a cheap and efficient way. Now an overview of characteristics, that will be present in general video streams from an urban intersection scene, is given:

- The roads leading to the intersections can have multiple lanes.
- Vehicles and pedestrians enter and leave the scene. This does not necessarily happen at the border of the image frame, for example a pedestrian can enter the scene by coming out of a building standing nearby the intersection.
- Queues of vehicles will often appear since traffic signs may be present. As a result vehicles will drive at very different speeds
- Cast-shadows and self-shadows of the vehicles and other objects might appear in the scene. Furthermore the shadows can disappear and reappear in a short period of time if the weather is cloudy.
- Also as a result of cloudy weather, sudden global illumination changes might occur often. A sudden illumination change also occurs at the moment when the street lights are turned off and on.
- Weather conditions like snow, fog or rainfall causes more noise in the video stream and makes the movement of the objects less detailed.
- Since we have an urban setting, not only cars and trucks will drive on the roads but also more versatile vehicles like bikes and scoot mobiles could be present. Also animals and pedestrians might cross the roads.
- Not only vehicles and pedestrians move in the scene but also environmental movements (depending on wind speed) like branches of nearby trees, flags, debris on the road, moving commercial billboards, etc.
- Vehicles can become part of the background when for example a car is parked in the scene.
- In general, urban intersections are very well lit, So detecting objects at night times should not be a more difficult operation then detecting objects at daytimes.

2.2. The tasks and requirements of a robust urban traffic monitor system

Given the setting in the previous section, the requirements of an urban traffic monitor system can be determined. First of all, the system should have an adaptive background model since we have to cope with (sudden) changes of illumination in the scene and objects can become part of the background. Secondly, since queuing of vehicles does occur often, the detection algorithm should successfully detect all separate vehicles, even

if they are very close to each other, which is the case with traffic queuing. This also requires that the algorithm should cope with queuing vehicles that have speed zero. These vehicles should clearly not be counted as background since they belong to the active traffic that we want to monitor. Thirdly, moving shadows should be detected and eliminated by the system. This helps in the vehicle segmentation step since shadows can cause errors on density estimation and the unwanted merging of vehicles might occur. Fourthly, a number of traffic parameters should be extracted from the video streams. By tracking the vehicles in the scene, it is possible to calculate parameters like speed, average traffic density, queue lengths and times for each lane. Fifthly, The information from each stream in the system should be combined in such a way that a visual flow of traffic can be made in real-time. This flow can be recorded and stored in a cheap and efficient way. The most important component in the system will be the background modelling system [6]. Operations like tracking, counting and segmentation of vehicles are not possible without a robust background system. For this reason, this Master thesis has its focus on this component.

3. System overview

In this section an overview of the components is given that should be present in a successful vision based urban traffic monitoring system. Each component from this framework will be described in some detail here. The framework should consist of:

- Multiple cameras at different intersections producing video streams
- Camera initialisation and calibration by defining interesting regions in the image frames for each camera and set a distance correction for each camera (geometry)
- *For each stream:*
    - A component that performs background subtraction and that eliminates shadows
    - A component capable of tracking each foreground object in the stream
    - A component capable of extracting traffic parameters for each foreground object in the stream
- A component recording the streams when necessary (video summarization)
- A central component which combines and stores traffic parameters from multiple streams
- A GUI showing the flow of the traffic and which can query upon the gathered data

In figure 2 on page 7, the relation between the components of the system is sketched, where arrows represent the direction of the processing steps. The boxes that are present in the red square are components that belong to the background subtraction component. Now all the components defined in this framework (figure 2) will be explained in some more detail:

Camera units – We aim here at using cheap low-resolution greyscale or colour cameras which can be compared to simple webcams. These cameras produce frames at a fixed frame rate which are the input data for the system.

Camera initialisation and calibration – In the initialisation step, interesting points in the static scene should be manually marked once. Interesting points could be areas where new vehicles can appear or disappear. Another possibility is to mark only the roads themselves as interesting areas if you are only interested in vehicles on the road. Calibration of the camera means that the relative transition from image coordinates to

world coordinates is known. This is important for the system to derive speeds and locations of vehicles in the scene.

Background subtraction - This component is responsible for generating a background model and a binary mask that displays all foreground objects in the current frame. Also incorporated in this component is the detection and elimination of shadows. This research emphases on these features. In the next chapter the steps in background subtraction will be described together with existing algorithms and methods used in literature.

Tracking foreground objects – The foreground objects found by the background subtraction in each frame should be tracked through time. This part is responsible for identifying and labelling the foreground objects in successive frames.

Update traffic parameters – This component should gather information about each labelled (and tracked) foreground object, such as speed, location, acceleration, size, type of object, etc. This information is then send for each frame to the central processing.

Video Summarization – For later analyses of the traffic at intersections this component is responsible for recording video only when interesting events at the intersections occur. So for example when there is no traffic at all, no video needs to be recorded.

Central processing – Receives traffic parameters from all available streams and stores them in a database for easy and fast retrieval. Parallelization is of great importance since data from different streams arrive simultaneously.

GUI – The GUI is responsible for letting the operator calibrate and initialize the cameras. Furthermore is should display the flow of traffic through the city. So at the intersections, information about the vehicles should be displayed by retrieving the traffic parameters from the central processing database.

4. Literature on background subtraction algorithms

Most of the background subtraction algorithms found in literature have a generic flow which consists of four stages [17]: pre-processing, background modelling, foreground detection and data validation. Each of these steps will be discussed in some detail in the next section. For traffic monitoring, eliminating shadows in video frames is also very important. Therefore, this step of shadow detection and elimination is incorporated in the background subtraction component.

4.1. Pre-processing

In the pre-processing step the raw data stream coming from the camera is processed in such a way that the format can be handled by the visual-based components of the system. In this early stage of processing, techniques are often used to reduce camera noise using simple temporal or spatial smoothing. Smoothing can also be used to eliminate environmental noise such as rain and snow captured by the camera. When applying a spatial smoothing, all the pixels in one frame are smoothed based on the values of surrounding pixels. In this way the image softens. When temporal smoothing is applied, the value of a pixel at a particular location (and surrounding locations) is averaged with the same pixel(s) from the next or multiple next/past frames. This tends to smooth motion.

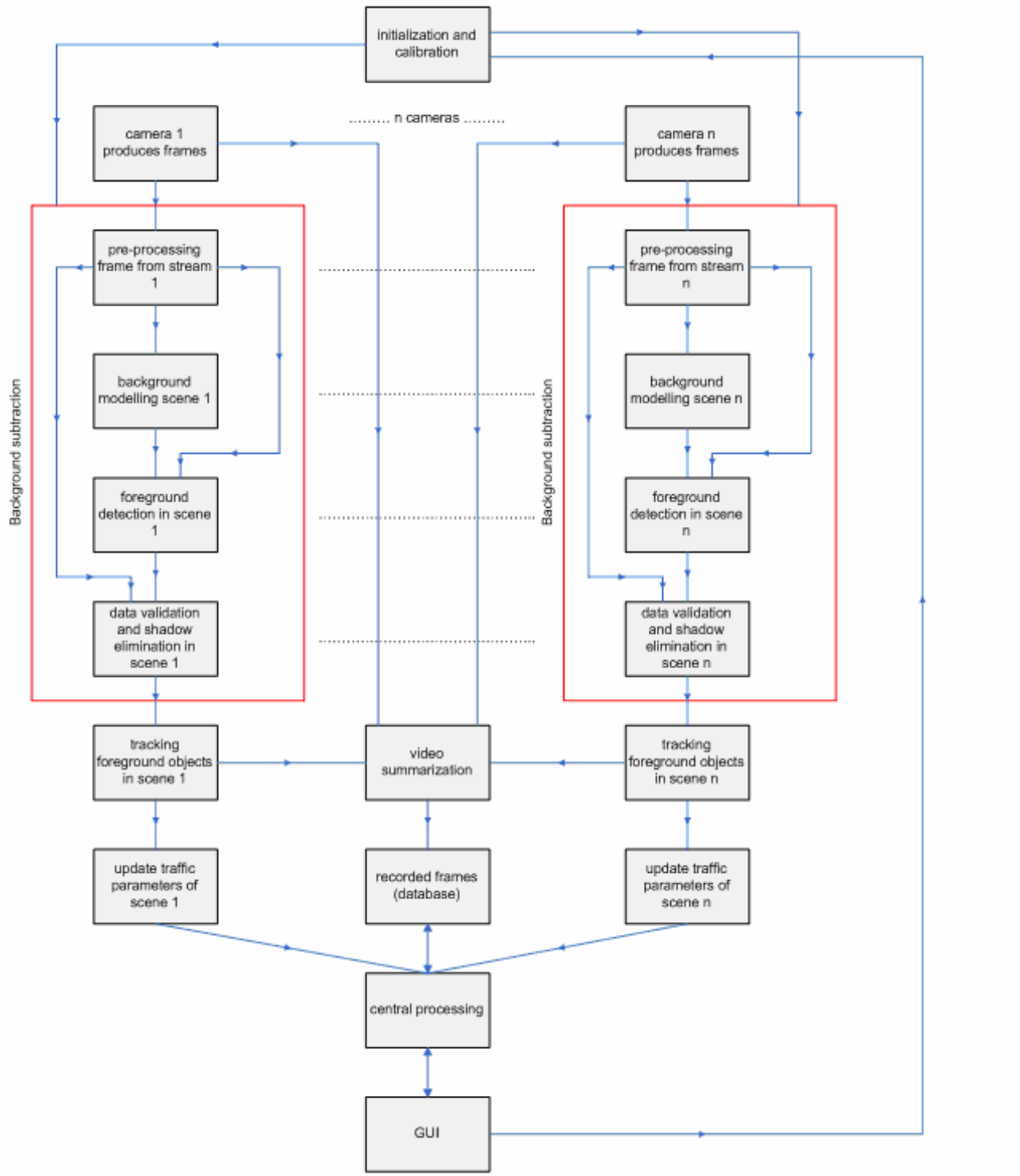

initialization and calibration
camera 1 produces frames
..... n cameras .....
camera n produces frames
pre-processing frame from stream 1
pre-processing frame from stream n
background modelling scene 1
background modelling scene n
foreground detection in scene 1
foreground detection in scene n
data validation and shadow elimination in scene 1
data validation and shadow elimination in scene n
Background subtraction
Background subtraction
tracking foreground objects in scene 1
video summarization
tracking foreground objects in scene n
update traffic parameters of scene 1
recorded frames (database)
update traffic parameters of scene n
central processing
GUI

Figure 2 *– Schematic overview of a general traffic monitoring system*

To make sure a traffic system runs in real-time, the frame rate can be reduced in this stage by not processing each frame. Also a reduction of camera resolution is possible by rescaling each incoming image frame. Another important issue in this first stage is how to store the data of the image frames from the stream. Some algorithms only handle luminance intensity which requires only one scalar value for each pixel in the image. In this case only a "black and white" video stream is needed. However, when using information of each colour channel in coloured frames, better subtraction results can be obtained [9,14], especially when objects in the scene are present in low-contrast areas. Modern cameras equipped with an CCD sensor linearly transforms infinite-dimensional spectral colour space to a three-dimensional RGB colour space via red, green and blue colour filters [11]. As a result of this linearity and sensor type, the following characteristics of the output images hold:

- *Variation in colour*: It rarely occurs that we observe the same RGB colour value for a given pixel over a period of time
- *Band unbalancing*: Cameras typically have different sensitivities to different colours. A possible approach taken is to balance the weights on the three colour bands by performing a normalization [11]. The pixel colour is normalized by its standard deviation s_i which is given by

$$s_i = [\sigma_R(i), \sigma_G(i), \sigma_B] \tag{1}$$

  where $\sigma_R(i)$, $\sigma_G(i)$ and $\sigma_B(i)$ are the standard deviation of the i^{th} pixel's red, green and blue values.
- *Clipping*: The CCD sensors have limited dynamic range of responsiveness. All visible colours lie inside the RGB cube spanned by the R,G and B vectors with a given range. As a result some pixel values (outside the range) are clipped in order to lie entirely inside the cube. To reduce this unusual shape of the colour distributions s_i is often set to a minimal value if a pixel's s_i is zero.

Also in the pre-processing step it can be advantageous to transform from the original RGB colour space to an invariant colour space. One such system is the normalized RGB colour space defined as:

$$r = \frac{R}{R+B+G} \qquad g = \frac{G}{R+B+G} \qquad b = \frac{B}{R+B+G} \tag{2}$$

where R,G,B are the values for red, green and blue coming directly from the camera, and r,g,b are the resulting normalized RGB colour components. This colour space has the important property that they are not sensitive to surface orientation, illumination direction and illumination intensity [20]. A disadvantage is that normalized colours become unstable when the intensity is small.

A more intuitive colour model is the HSI colour system, which defines a value of hue, saturation and intensity for each pixel value. The values H,S and I are obtained from the R,G and B values in the following way [20]:

$$H = \frac{\sqrt{3}(G-B)}{(R-G)+(R-B)} \qquad S = 1 - 3 * \min(r,g,b) \qquad I = \frac{(R+B+G)}{3} \tag{3}$$

By taking a linear combination of H and S values, an invariant model can be intuitively constructed. Another popular invariant colour model, is the $c_1c_2c_3$ model. (see , [12] and [21]) This model is proven to be invariant to a change in viewing direction, object geometry and illumination. The features are divined as:

$$c_1 = \arctan(\frac{R}{\max(G,B)}) \qquad c_2 = \arctan(\frac{G}{\max(R,B)}) \qquad c_3 = \arctan(\frac{B}{\max(R,G)}) \tag{4}$$

4.2. Background Modelling

In the background modelling stage, not only a model of the background of the scene should be constructed once, but also the model should be adaptive. The general aim of the background modelling component is therefore to construct a model that is robust against environmental changes in the background, but sensitive enough to identify all relevant moving objects. Background modelling techniques can be classified into two broad categories [17]: non-recursive and recursive. (these categories will be introduced shortly) The methods described next, all assume a highly-adaptive modelling, which means that the model is updated using only a fixed *small* number of history frames. In this way current incoming frames have a great influence on the final model: it can adapt quickly.

In non-recursive background modelling techniques [2,3,5,14,22], a sliding-window approach is used for background estimation. A fixed number of frames is used and stored in a buffer. With a sliding-window size n, the background model $B_t(x,y)$ at a certain time t is given by:

$$B_t(x,y) = F(I_{t-1}(x,y),...,I_{t-n}(x,y)) \tag{5}$$

where $I_t(x,y)$ is the image frame coming from the camera at time t and F is a function that is based on the temporal variation of pixel values and can be obtained in different ways as given in literature:

- *Frame differencing*: This is the most easy and intuitive background modelling method. It uses the video frame at time $t-1$ as the background model for the frame at time t. The background model at time t can be summarized as:

  $$B_t(x,y) = I_{t-1}(x,y)\,. \tag{6}$$

- *Median filter*: This is one of the most commonly used background modelling techniques. In a greyscale video stream, the background is estimated using the median at each pixel location of all the video frames in the buffer of length n:

  $$B_t(x,y) = median(I_{t-1}(x,y),...,I_{t-n}(x,y)) \tag{7}$$

  Median filtering can also be applied on coloured video frames. Instead of taking

the median, [22] uses the mediod, which takes into account both R, G and B channels of the video frame.

- *Linear predictive filter*: In this method, the current background estimate $B_t(x,y)$ is computed by applying a linear predictive filter on the pixels in the buffer of size `n`. A predictive filter uses information of history frames to predict the values in future frames. In the setting of a background modelling algorithm, a linear predictive filter can be written as follows:

$$B_t(x,y) = \sum_{i=1}^{n} \alpha_i I_{t-i}(x,y) \tag{8}$$

  where α_i is the predictive coefficient at frame `i` in the buffer. A disadvantage of this method is that coefficient α (based on the sample covariances) needs to be estimated for each incoming frame, which makes this method not suitable for real-time operation.

- *Non-parametric model:* This method discriminates itself from the mentioned methods above, by not using a *single background estimate* $B_t(x,y)$ at each pixel location, but using *all the frames in the buffer* to estimate a non-parametric estimate of a pixel density function. (see [14]) A Gaussian kernel estimator is usually used to model this background distribution. A pixel can be declared as foreground if it is unlikely to come from the formed distribution. The advantage of using non-parametric models for estimation, is that it can correctly identify environmental movements like moving tree branches (which are undesired foreground objects).

Recursive background modelling techniques do not keep track of a buffer containing a number of history frames. Instead, it recursively updates a single background model based on each incoming frame. These techniques are used a lot in literature. (see [15,16,17,26,27,28]) Recursive techniques require less storage than non-recursive techniques but an error that might occur in the background model will be visible for a long time. Now three representative recursive techniques for background modelling found in literature will be described shortly:

- *Approximated median filter*: Since the good results that can be obtained using non-recursive median filtering, a approximated recursive variant was proposed. This method is particularly interesting since it is reported that this method is successfully applied in an traffic monitor system [17]. With this approach, the running estimate of the median is incremented by one if the input pixel is larger than the estimate, and decreased by one if the pixel is smaller than the estimate. This estimate converges to a value for which half of the input pixels are larger than and half are smaller than this value, which is exactly the same that we get when calculating the median in the non-parametric approach.

- *Kalman Filter*: In its most general form, a Kalman filter is an efficient recursive filter which estimates the state of a dynamic system from a series of incomplete or noisy measurements. Applied to background modelling, it should "estimate the state" of the current background model. The Kalman filter should learn from its errors made in the past and the rate of uncertainty should be minimized. A number of versions of Kalman filters are used in literature. The simplest version

uses only the luminance intensity but more complex versions also take into account the temporal derivative or spatial derivatives. The internal state of the system is then described by the background intensity B_t and its derivative B'_t. These two features are recursively updated using a particular scheme [17]. In general, background modelling using Kalman filters performs not very well, since the model is easily affected by the foreground pixels, causing long trails after moving objects [15,17].

- *Mixture of Gaussians (MoG)*: The MoG method tracks multiple Gaussian distributions simultaneously, unlike the Kalman filter which tracks the evolution of only a single Gaussian. This method is very popular since it also capable of handling multi-modal background distributions as described above at the non-parametric approach. Similar to this non-parametric model, MoG maintains a density function for each pixel. On the other hand MoG is parametric, the model parameters can be adaptively updated without keeping a large buffer of video frames. [17,26,27,28]. The disadvantages of this model is that it is computationally intensive and its parameters require careful tuning. It is also very sensitive to sudden changes in global illumination: it can happen that the entire video frame will be classified as foreground. Since one of the implementations in this research will use a MoG model, this method is described in more detail in chapter 5.2.

4.3. Foreground detection

The goal of the foreground detection component is to obtain a binary mask image $M_t(x,y)$ for each video frame. In this mask only objects that belong to the foreground, will be visible. Basically, in this step, it should be determined which pixels in the current image frame are candidates of being foreground pixels. The images required for this component are the current background model $B_t(x,y)$ (obtained in the component described in 4.2) and the current incoming image frame from the camera $I_t(x,y)$ after pre-processing. The most commonly used approach for foreground detection is to check whether the input pixel is significant different from the corresponding background estimate. The creation of a binary mask $M_t(x,y)$ for a greyscale video frame with height `h` and width `w` can be described by:

```
for x = 1 to w
  for y = 1 to h
    if |I_t(x,y) − B_t(x,y)| > T then M_t(x,y) = 1; else M_t(x,y) = 0      (A1)
  endfor;
endfor;
```

where T is a threshold value. In most cases the value for T is determined experimentally and therefore depends strongly on the setting of the scene. Another popular foreground detection scheme is to threshold based on the normalized statistics. In that case $|I_t(x,y) - B_t(x,y)| > T$ in the pseudo-algorithm above should be replaced with

$$\frac{|I_t(x,y) - B_t(x,y) - \mu|}{\sigma} > T \qquad (9)$$

which is called the z-score, where μ and σ are the mean and the standard deviation of $I_t(x,y) - B_t(x,y)$ for all neighbouring pixels around (x,y). Although in general better results are obtained, this step is computational more expensive since it requires an extra

scan through both the current image frame and the current background model to obtain the mentioned mean and standard deviation. Instead of T being a scalar, it would be ideal to let T be a function of the spatial location (x,y). For example, in regions of low contrast the threshold should be lower since pixel values of the background and relevant foreground objects are more close together in that case. Another approach is to use the relative difference rather than absolute difference to emphasize the contrast in dark areas such as shadows [23]. In that case the equation will be:

$$\frac{|I_t(x,y) - B_t(x,y)|}{B_t(x,y)} > T \tag{10}$$

A drawback of this method is that it cannot be used to enhance contrast in bright images such as an outdoor scene under heavy fog. There are also approaches where *two* thresholds are used [22]. The basic idea is to first identify "strong" foreground pixels whose absolute differences with the background estimates exceeded a large threshold. Then, foreground regions are *grown* from strong foreground pixels by including neighbouring pixels with absolute differences larger than a smaller threshold. This process is called hysteresis thresholding.

4.4. Data Validation

The goal in this step of the background subtraction phase, is to improve the results we obtained in the form of a binary mask (chapter 4.3). The background models described in chapter 4.2. have these general drawbacks [17]:

1. All models ignore any correlation between neighbouring pixels
2. The rate of adaption may not match the moving speed of the foreground objects. This is especially the case in an urban setting, since cars with all kinds of speeds will pass by.
3. In most of the models, Non-stationary pixels from moving leaves or shadow cast by moving objects are easily mistaken as true foreground objects.

As a result of the first drawback, a lot of small false positives and false negatives will occur distributed across the candidate mask. One solution for this problem is to discard all classified foreground objects which have such a small size that the object can never be a vehicle or pedestrian. For this a connected component grouping algorithm can be used. Using morphological filtering, like dilation and erosion on foreground masks, eliminates isolated foreground pixels and merges nearby disconnected foreground regions. As a result of the second drawback, it can happen that large areas of false foreground (also called "ghosts") might be visible in the binary mask [22]. This happens when the background model adapts at a slower rate than the foreground scene. A solution to this problem is to use multiple background models running at different adaptation rates, and periodically crass-validate between different models to improve performance [14]. A drawback of this multiple background modelling and the cross-validation is that it is very computational expensive. Furthermore in an urban setting, objects move at a very different speeds, which results a large number of different background models. Finally, to come up with a solution for the third drawback is very important since especially in urban traffic monitoring we are interested in vehicle classification and road density (what percentage of the road is occupied on average?). To obtain representative measurements for these features, it is necessary to remove all shadows from moving candidate objects in the image frames. In the next section, shadow elimination will discussed.

4.5. Shadow detection and elimination

Figure 3 *– cast-shadows can overlap other foreground objects*

The goal of a successful shadow elimination algorithm is to prevent moving shadows being misclassified as moving objects or parts of them. If shadows are not eliminated it could happen that two separate objects will be merged together by the system when the shadow of one object is overlapping with the second object. A shadow can be classified as a self-shadow or a cast-shadow. A self-shadow is the part of the object which is not illuminated directly by the light source. In our case of traffic monitoring the relevant light sources are the sun, street lights and vehicle lights. The cast-shadow of an object is the area projected on the scene by the object. This area can be very large especially in the sunset and sunrise situations (see figure 3). A cast-shadow can be further classified in umbra and penumbra. When an object is fully opaque, so all the light is blocked directly by the object, the cast-shadow that occur is a umbra. When the object is partially transparent only a certain amount of light is blocked and the cast shadow that occurs is a penumbra. Looking at the setting of traffic monitoring and its segmentation step, it can be seen that cast-shadows are important to eliminate. Self-shadows should not be eliminated since they will result in incomplete object silhouettes [25].

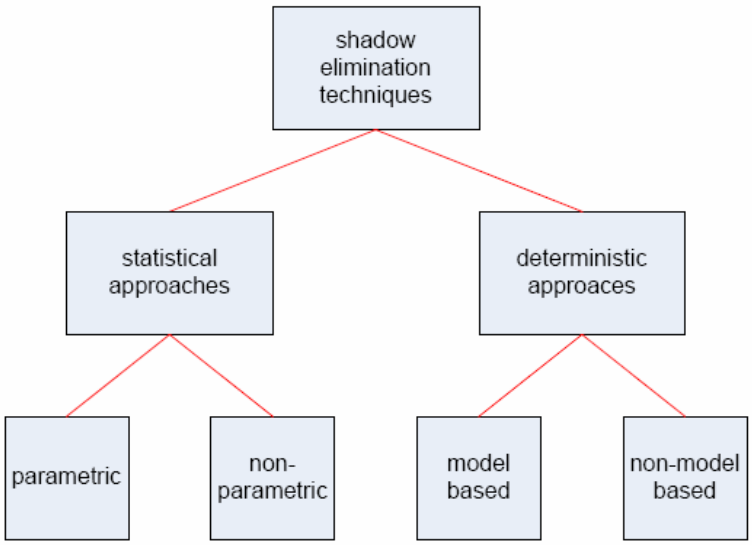


Figure 4 *– Types of shadow elimination techniques, as stated in [10]*

In figure 4 an hierarchy of shadow elimination techniques in literature is displayed. The two main approaches often taken are *the deterministic approach* [7,8,12,14,24] and the *statistical approach* [9,11,25]. In the former approach the decision is just a binary decision process, a pixel just belongs to the foreground object or it belongs to its cast-shadow. In the latter approach uncertainty is added by using probabilistic functions to describe the class membership. Adding uncertainty to the class membership assignment can reduce noise sensitivity [13].

The deterministic approaches can be further subdivided. *Model based deterministic* approaches rely on the notion that the luminance ratio (the ratio of the intensity value of

a pixel when it is under shadow to that when it is under illumination) is a constant [24]. A linear transformation is used to describe the reduction of pixel intensity in shadow regions. When a new image frame comes in, its pixels with illumination reductions that follow the linear model are then identified as probable shadow pixels. *Non-model based deterministic* approaches use a comparison between the current image frame `I(x,y)` and the background image frame (obtained from an arbitrary model) `B(x,y)` and then set a threshold to classify a pixel as a shadow or non-shadow pixel, for example the Sakbot system [10], which works in the HSV colour space, uses the following equation to determine if a pixel is a shadow pixel:

$$SP(x,y) = \begin{cases} 1 : \alpha \le \dfrac{I^V(x,y)}{B^V(x,y)} \le \beta \wedge (I^S(x,y) - B^S(x,y)) \le T_s \wedge \left| I^H(x,y) - B^H(x,y) \right| \le T_H \\ 0 : otherwise \end{cases} \tag{11}$$

where $I^c(x,y)$ and $B^c(x,y)$ with `c={H,V,S}` is the hue, saturation and value component of the pixel In this equation α takes into account the power of the shadow, that is, how strong the light source is with regard to the reflectance of the objects. So in sunny outdoor scenes a low value for α should be set. β prevents that background changes due to camera noise are not classified as shadow. T_S and T_H are thresholds for the saturation and hue component and its goal is similar as described in chapter 4.3.

The statistic approaches can also be divided further. In the statistical methods the parameter selection is critical issue, so we have *parametric* and *non-parametric* statistical approaches. An example of the latter approach is where colour is considered as a product of irradiance and reflectance [11]. The distortion of the brightness and chrominance of the difference between the *expected colour* of a pixel and its value in the current image are computed. The rationale used is that shadows have similar chromaticity, but lower brightness than the background model. A statistical learning method might be used to automatically determine the thresholds for the components of the expected colour [11]. Finally in the parametric statistical approach often two sources of information are used to help detecting shadows and objects: local information (individual pixel values) and spatial information (objects and shadows are compact regions in the scene) For example the Anton system [10,13] uses the local information in the way that if v = $[R,G,B]^T$ is the value of the pixel not shadowed, a shadow changes the colour appearance by means of an approximated linear transformation:

$$\bar{v} = \begin{pmatrix} 0.48 & 0 & 0 \\ 0 & 0.47 & 0 \\ 0 & 0 & 0.51 \end{pmatrix} v = Dv \tag{12}$$

The authors also note that in traffic video scenes the shadows often appear more bluer. Given the means and variances for the three colour channels for reference points, the shadowed versions of means and variances will then become $\mu^i_{SH} = \mu^i_I d_i$ and $\sigma^i_{SH} = \sigma^i_I d_i$ with $i \in R,G,B$. The spatial information is extracted by performing an iterative probabilistic relaxation by using information of neighbourhood pixel values. A drawback of parametric approaches is that the parameters need to be selected very carefully.

Manual segmentation of an initial number of frames is often needed to obtain good learning results. An expectation maximization (EM) algorithm can be used to automatically find these parameter values.

while developing and evaluating a shadow detector three important quality measures are very important to consider [10]. First of all good detection is needed (low probability of classifying shadow points as non-shadow points). Secondly the probability to identify wrong points as shadow should be low (*good discrimination*) and finally the points marked as shadows should be as near as possible to the real position of the shadow point (*good localization*). Prati *et al.* performed an evaluation of the general types of shadow elimination described in the previous paragraph [10,13]. They came up with some important conclusions. When a general-purpose shadow detector is needed with minimal assumptions, a pixel based deterministic non-model-based approach assures best results. To detect shadows in one specific environment, more assumptions yield better results and the deterministic model-based approach should be used. If the number of object classes becomes to large when considering a model based approach, it is always better to use a complete deterministic approach. For indoor environments the statistical approaches are the more reliable since the scene is constant and a statistical description is very effective. Conclusions of Prati *et al.* are that a statistical parametric spatial approach better distinguishes between moving shadows and moving objects (in contrast to deterministic approaches) and a deterministic non model based approach detects moving shadows better (in contrast to statistical approaches). So a deterministic non-model based approach gives good detection while a statistical parametric approach gives good discrimination.

5. Implementations

This chapter describes the two implemented background subtraction algorithms in this research project (5.1 and 5.2). The implementation of two shadow detectors will be discussed in 5.3, one suitable for the first subtraction algorithm and one for the other. In 5.4 we describe how we applied a simple video summarization to obtain compact videos only containing moving traffic. Although the focus of this research is not on video summarization, it is a nice application to test the two implemented background subtraction algorithms. Finally 5.5 describes the steps to actually work with the created software and its wrapper. The software constructed in this research is written using Microsoft Visual C++ 6.0 and the Open Source Computer Vision Library (OpenCV[1]).

5.1. Deterministic background model

The algorithm described in this section is based on the methods defined in [1] and [4]. As a prerequisite for this algorithm N frames should be available which represents a perfect background of the static scene. So in our setting we want N frames of the intersection when no traffic is present on the roads. The first step of the algorithm is to take a pixel-wise average or median of the `N` frames that will result in the initial background model. If the video frames have a resolution of `w` x `h`, then the initial background model $B(x,y,c)$ with c $\in$ {R,G,B}, calculated from mean values, will be:

[1] Website of the Open Source Computer Vision library: http://www.intel.com/technology/computing/opencv/

```
for x = 1 to w
  for y = 1 to h
    for c = 1 to 3
```
$$B(x,y,c) = \frac{1}{N}(I_1(x,y,c) + ... + I_N(x,y,c)) \quad \text{[A2]}$$
```
    endfor;
  endfor;
endfor;
```

As said before, it is also possible to take the median of the N frames as the initial background model. This can be specified by the user by changing a command line parameter (see 5.5). The advantage of taking the median, is that an outlier pixel within the N frames is filtered out and does not contribute in the final initial background model. In case the median is used, N should be odd and the background model $B(x,y)$ will be:

```
for x = 1 to w
  for y = 1 to h
    for c = 1 to 3
      vector v = [ I_1(x,y,c)  I_2(x,y,c) ... I_N(x,y,c) ]   [A3]
     order pixel values in v from low to high;
     B(x,y,c) = v[(N+1)/2];
    endfor;
  endfor;
endfor;
```

When the initial background model is created the background subtraction process starts. For each new frame $I(x,y)$ that comes in, it is first subtracted from the current background model using the first pseudo-algorithm given in chapter 4.3 [A1] (adjusted for colour instead of greyscale frames). The value for the threshold parameter T should also be given by the user as a command line parameter. This subtracting results in a binary mask where foreground objects are visible in white and the background in black. The next step in the process is to update the current background. Only all the pixels that are classified as *background* in the mask $M(x,y)$ are updated in the background model $B(x,y)$ in the way described in the pseudo algorithm [A4].

```
for x = 1 to w
  for y = 1 to h
    if M(x,y) == 0 and |I(x,y,R) - B(x,y,R)| < α and |I(x,y,G) - B(x,y,G)| < α
                   and |I(x,y,B) - B(x,y,B)| < α then
```
[A4]
$$B(x,y) = \frac{1}{3}(2 * B(x,y) + I(x,y))$$
```
  endfor
endfor
```

where $\alpha = T/10$ which is determined experimentally. The concatenation of $|I(x,y) - B(x,y)| < \alpha$ conditions is needed to avoid the creation of clusters of foreground pixels which are in reality just background. Figure number 5 demonstrates how these clusters are formed when the mentioned conditions are *not* used. Assume that the black

area (background) has pixel values of 0. The white area (object of interest) has pixel values of 70 and the grey area (object shadow and/or noise artefacts) has values of 35. Now also assume that the value for T (in A1) is set to 60. Now we monitor a single pixel in a number of frames, coloured red in figure 5. In figure 5A the pixel is clearly classified as background and thus the pixel at the same location in the background model is updated and (again) set to 0. In figure 5B, one frame later, the pixel of interest is again classified as background (35 < T) and the corresponding pixel in the background model is now set to 35/3. In the next frame (figure 5C), the pixel is *still* classified as background since (70-(35/3)) < T. As a result of this error, the corresponding pixel in the background model will be updated with a higher value. Now suppose that the object stands still for a number of frames (in situation c). Every time the pixel will be classified as background and the value for the background pixel in the model will become higher and higher. Now suppose the object moves on after a while (situation d). `I(x,y)` will be 0 but the corresponding pixel in the background model has become > T. Therefore the pixel in situation d will be classified as a foreground pixel. Pixels in the neighbourhood of our pixel of interest also have a high probability to undergo the same process as described. In this way clusters of undesired foreground pixels will occur.

Figure 6 shows an example of the effect of not including the concatenation of $|I(x,y)-B(x,y)|<\alpha$ conditions. A car entered the scene from the bottom right and clearly a trail of unwanted foreground pixels is visible.

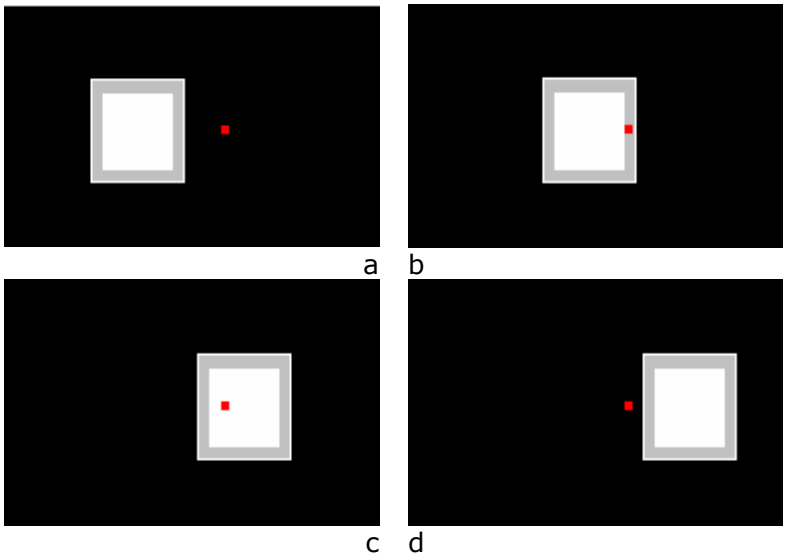


Figure 5 *– The creation of a unwanted foreground pixel*

Figure 6 *– trail of unwanted foreground pixels as result of a car that passes by*

By updating the background in the way described above, the global lightning changes through time can be handled correctly. To clarify and summarize this algorithm, a flowchart of this algorithm is displayed in figure number 7.

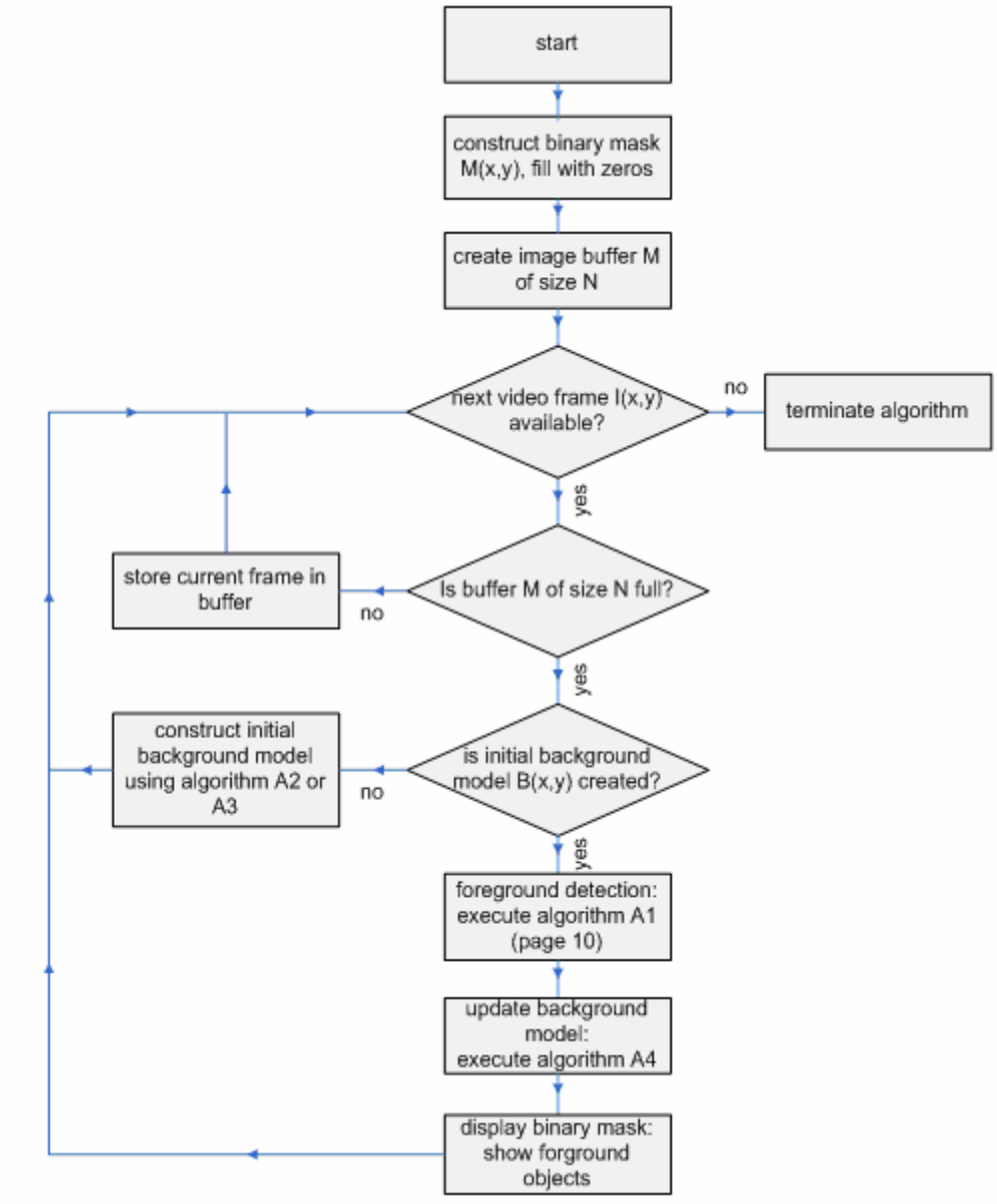

start
construct binary mask M(x,y), fill with zeros
create image buffer M of size N
next video frame I(x,y) available?
no
terminate algorithm
yes
store current frame in buffer
Is buffer M of size N full?
no
yes
construct initial background model using algorithm A2 or A3
is initial background model B(x,y) created?
no
yes
foreground detection: execute algorithm A1 (page 10)
update background model: execute algorithm A4
display binary mask: show forground objects

Figure 7 *– flowchart of the background subtraction implementation (deterministic approach)*

Both the background model and mask are stored using the `IplImage` datatype available in OpenCV. Operations on the images are efficiently executed when this datatype is used. To identify the separate foreground objects, a library, `cvBlobsLib`[1], is used to display bounding boxes around foreground objects. The display of bounding boxes can also be disabled by the user with a command line parameter.

[1] Website of the cvBlobs Library: http://opencvlibrary.sourceforge.net/cvBlobsLib

5.2. Statistical background model (mixture of Gaussians)

The second implemented background subtraction system is based on a statistical approach proposed by [26] and elaborated further by [28]. For each pixel in the scene a Gaussian Mixture Model (GMM) is considered. At any time, t, what is known about a particular pixel, `{x0, y0}`, is its history: $\{X_1,...,X_t\} = \{I(x,y,i) : 1 \le i \le t\}$ where `I` is the image sequence. When making scatter plots of these history values, it becomes clear that a GMM is needed to construct an accurate background model. In figure 8 a (R,G) scatter plot is showing a bi-model distribution of a pixel's value over time resulting from CRT monitor flickering [26]. It can be seen that two clusters of points are formed. This is already an indication that a single model (Gaussian) is not sufficient to create a reliable background model. Another interesting aspect is what to do with the pixel value history. For example the value of a certain pixel in a scene might change abruptly for a long time when it becomes cloudy at daytime. As a result, multiple clusters will also occur in the long-term scatter plot. So it can be concluded that not all the pixel values in the history should be used and more recent observations should get a higher weight.

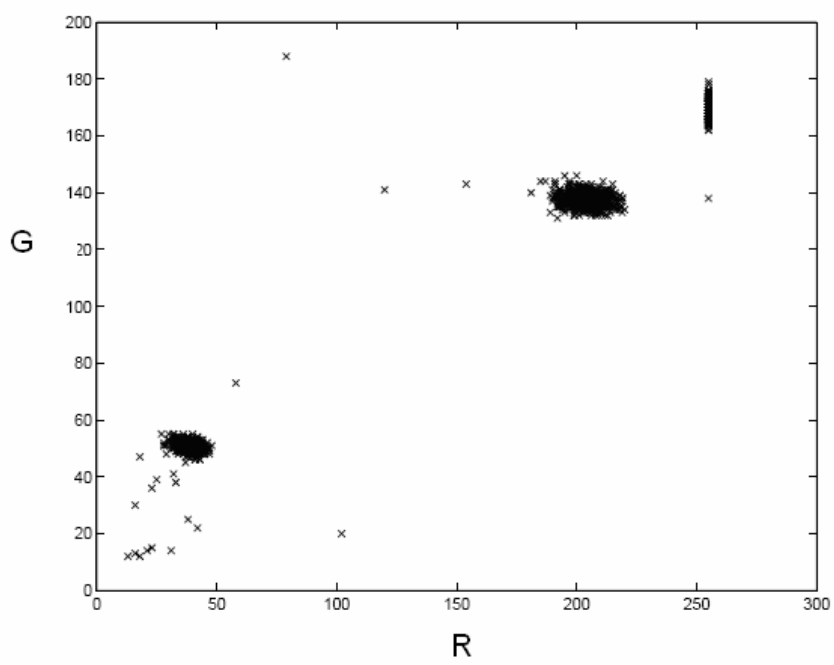


Figure 8 *– (R,G) Scatter plot of one particular pixel measured over time.*

So the first step is to model the recent history $\{X_1,...,X_t\}$ by a mixture of K Gaussian distributions. The probability of observing the current pixel value can be written as follows [26]:

$$P(X_t) = \sum_{i=1}^{k} \omega_{i,t} \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \qquad (13)$$

Where `K` is the number of Gaussian distributions. $\omega_{i,t}$ is the weight of the i'th Gaussian at time `t`. $\mu_{i,t}$ and $\Sigma_{i,t}$ are the mean and covariance matrix of the i'th Gaussian in the mixture at time t. For computational reasons, $\Sigma_{i,t}$ is of the form $\sigma^2 I$ which assumes independence of the colour channels and also the same variance for each channel. η is a standard Gaussian distribution function of the form:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1}(X_t-\mu_t)} \quad (14)$$

The value for K, the maximum number of Gaussian components, depends on the amount of memory present. In the implementation K is set to a fixed value of 4. n is the dimension of the data. In case of grey value images this value is 1, if the image is in colour, n should be 3.

A new incoming pixel will most likely be represented by one of the major components of the mixture model. If this is the case, then the corresponding component in the mixture is updated using that new pixel value. So every new pixel X_t is matched against the K-Gaussian components until a match is found. A match can be defined as a pixel value that is within 2.5 standard deviations of a distribution [26]. In our implementation, the width of the band can be set by the user using a command line parameter. This can be done by setting a value for the squared Mahalanobis distance C. (see section 5.5). If none of the K distributions match the new pixel value, then the distribution with the lowest probability is replaced by a new distribution with a mean equal to the new pixel value. The variance of this new distribution is set to a high value and its weight ω to a relative low value.

At each pixel X_t in a new frame, the following update equations for the k-distributions are performed: [26]

$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(E_{k,t})$ with $E_{k,t}$ = 1 for the matching model, 0 for the remaining models (15)

α is the learning, rate, or rate of adaptation. This value, ranging from 0.0 to 1.0, determines the importance of the previous observations. Assigning high values to α results in the decrease of importance of these previous observations. This parameter should also be given by the user as a command line parameter. After this approximation, the weights are renormalized. The μ and σ parameters for unmatched distributions remain the same. The parameters of a distribution that matches the new observation is updated as follows:

$$\mu_{,t} = (1-\rho)\mu_{t-1} + \rho X_t \quad (16)$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \quad (17)$$

where $\rho = \alpha\eta(X_t \mid \mu_k, \sigma_k)$ (18)

Now that there is a GMM for each pixel location, a background model should be constructed from the GMM. To accomplish this, we look at those Gaussians in the mixture that are most likely produced by background processes. These Gaussians can be found by observing that these Gaussians have the most supporting evidence and lowest variances [26]. Therefore the authors order the K distributions in the mixture by the value of ω/σ. This value increases both as a distribution gains more evidence and as the variance decreases. Then finally the first B distributions are chosen as the background model, where:

$$B = \arg\min_b \sum_{k=1}^{b} \omega_k > T \tag{19}$$

The value of `T` determines the minimum portion of the data that should be accounted for by the background. When higher values for T are chosen, a multi-model background model is formed which can handle repetitive background motion. The limitations of the system described above is that the maximum number of Gaussian components per pixel is low and a low-weight component is only deleted when this maximum number of components is reached. The implementation of the statistical background model in our research is analogue to the method described by Zivkovic (see [28]) which overcomes these limitations.

An important extension in the work of Zivkovic with respect to Stauffer *et al*. (See [26]) is the introduction of automatically selecting the number of required distributions, `K`. Initially the system starts with just one component in the GMM centred on the first sample. Zivkovic looks at the prior evidence for a class c_k, which is the number of samples that belong to that class a priori. Using this information, it is possible to exclude classes from the set of distributions. In this way components can be active or inactive dynamically. Components are removed as soon as their weights ω become negative. A more detailed description of this dynamic class size determination can be found in Zivkovic's work [28]. Since prior evidence was introduced by the authors, the update equation for the weight of a distribution will become:

$$\omega_{k,t} = \omega_{k,t-1} + \alpha(E_k^t - \omega_{k,t-1}) - \alpha(\frac{c_k}{T}) \tag{20}$$

where T is again the number of history elements in the history buffer and c_k denotes the prior evidence. As can be seen from equation (20) the distribution weights can now become negative. Zivkovic uses the same update equations for the mean and covariance of the distributions (equations 16 and 17).

For the implementation of this statistical approach the C++ library[1] from the author of [28] is used, which performs the background subtraction the way it was described above. A wrapper for this library is created in such a way that the basic command line parameters match those of the implementation of the deterministic background model described in 5.1.

To summarize the statistical background subtraction algorithm, a flowchart is depicted in figure 9 which describes the process of the system. Note that the number of samples per distribution is limited to the size of the buffer of history pixel-values that is used. The buffer operations are not included in the chart for including them makes the chart unnecessary complex. Also note that processes within the red bounding box are repeated for each pixel `(x,y)` in the current frame `I`. When the mask for the current frame is

[1] Link to the C++ library used for the implementation of the statistical approach: http://staff.science.uva.nl/~zivkovic/Publications/CvBSLib0_9.zip

created, that is, when all pixels of current frame `I` are processed, the algorithm continues at the arrow pointing out of the red bounding box.

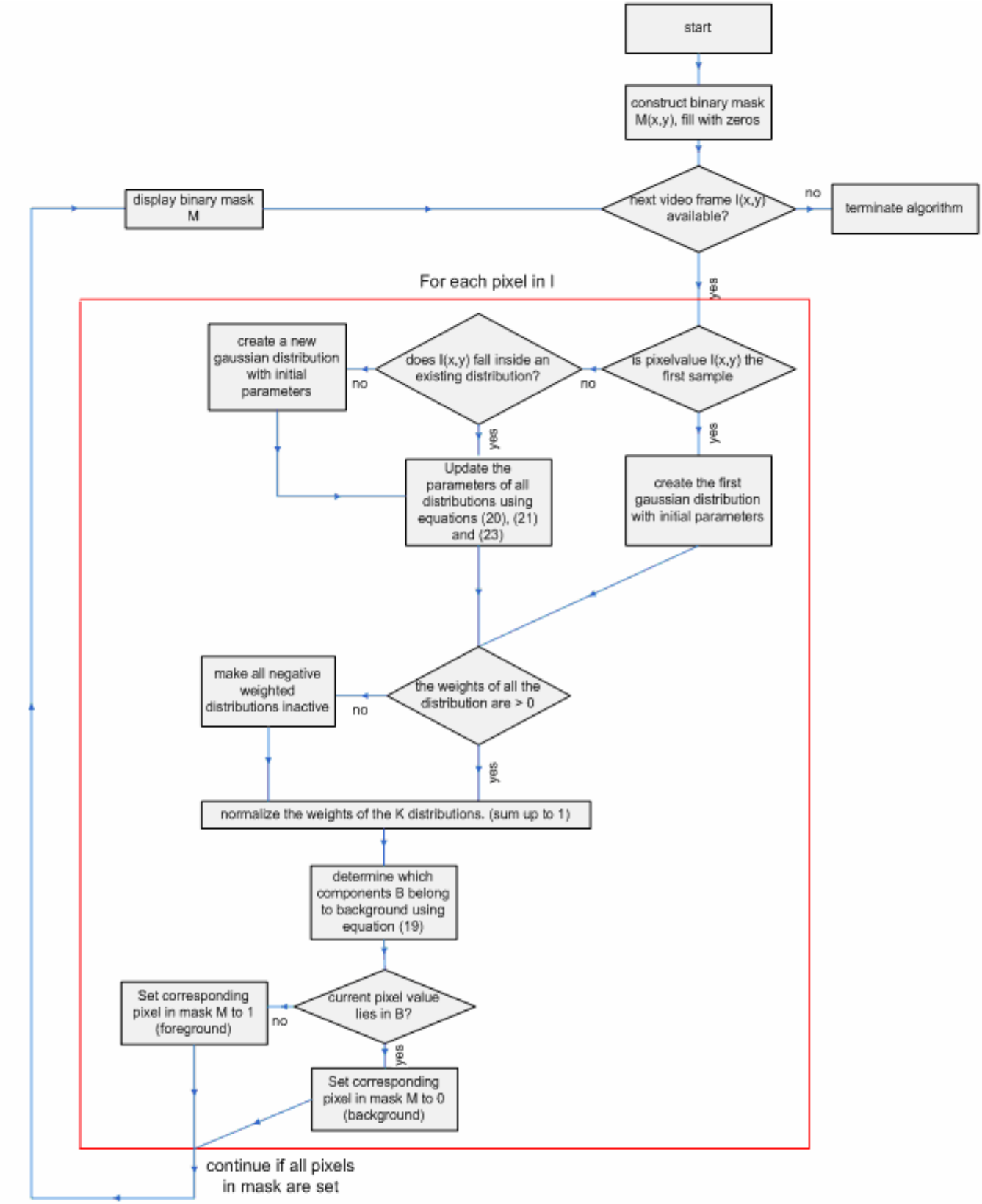

start
construct binary mask M(x,y), fill with zeros
next video frame I(x,y) available?
no
terminate algorithm
display binary mask M
For each pixel in I
yes
create a new gaussian distribution with initial parameters
does I(x,y) fall inside an existing distribution?
no
Is pixelvalue I(x,y) the first sample
no
yes
yes
Update the parameters of all distributions using equations (20), (21) and (23)
create the first gaussian distribution with initial parameters
make all negative weighted distributions inactive
no
the weights of all the distribution are > 0
yes
normalize the weights of the K distributions. (sum up to 1)
determine which components B belong to background using equation (19)
Set corresponding pixel in mask M to 1 (foreground)
no
current pixel value lies in B?
yes
Set corresponding pixel in mask M to 0 (background)
continue if all pixels in mask are set

Figure 9 *– flowchart of the background subtraction implementation (statistic approach)*

5.3. Shadow detection and removal

In this section the implemented shadow detectors are discussed. The focus is on moving shadows only, so only pixels that are classified as foreground by the background subtraction component could be candidates for shadow pixels. As mentioned in chapter 4.5 detecting moving shadows is very important for reliable segmentation and vehicle classification. (figure 3). Both background subtraction algorithms use an almost similar shadow detection algorithm. In the statistical setting, an algorithm is applied that reuses data of the Gaussian components to predict which foreground pixels are actually shadow pixels. This algorithm will be described in further detail in 5.3.2. For the deterministic approach, a shadow detector is implemented which is based on statistics gathered from pixel values of a number of history frames (see 5.3.1). The similarity between the two algorithms lies in the fact that both detectors classify a foreground pixel as shadow pixel when the pixel value has a significant lower value than it's corresponding background value.

5.3.1. Shadow detector for the deterministic background subtractor

In the implementation of the deterministic approach no statistical information is stored about the history of the pixels. For the correct working of the shadow detection algorithm a history of values for each pixel in previous frames is needed to derive the mean of the history values. The number of history frames to consider can be adjusted by the user as a command line parameter `-Sf`. We use the property stated in [28] that a pixel is classified as a shadow pixel when the pixel value has a significant lower value than it's corresponding background value (derived from the history frames). Also a parameter can be set by the user, τ, which is a threshold on how much darker the shadow can be. For example τ = 0.5 means that if the pixel is more than two times darker, then it is *not* shadow. The pseudo algorithm used for the shadow detector in the deterministic implementation is as follows:

- for the first `Sf` incoming frames: store the pixel values of these frames in a buffer `F`. shadow detection will be active after these `Sf` frames are stored.
- for each pixel that is initially classified as foreground pixel the following operations are executed to see if the pixel is a shadow pixel candidate:
    - calculate the R,G and B means and variances from the corresponding pixel value history stored in buffer `F`
    - from $c = (R, G, B)$ and $\mu = (\mu_r, \mu_g, \mu_b)$ calculate a ratio of similarity between the current pixel R,G and B values and their calculated means:

      $$S_{RGB} = \frac{c\mu^T}{\|\mu\|^2} \qquad (21)$$

      (see appendix A for a graphical derivation of equation 21)

      When the current pixel is more intensive than it's mean, S_{RGB} will be greater than 1. if the current pixel is darker S_{RGB} will always be between 0 and 1.
    - if S_{RGB} < 1 and S_{RGB} > τ then the current pixel is a *shadow pixel*. else the pixel is a pure foreground pixel
    - update mask at corresponding location (write value 255 for pure foreground pixel, write 125 for shadow pixel) and continue with next pixel in the current frame

5.3.2. Shadow detector for the statistical background subtractor

As mentioned in 5.2, the statistical background implementation uses the *cvBSlib*. This library has also a built in pixel based shadow detector which uses data from the underlying model of the pixel. As mentioned before, the basic idea of the shadow detector is that is classifies foreground pixels as shadow pixels when the pixel value has a significant lower value (darker) than it's corresponding background value (derived in this case from the background model of that pixel) [28]. Also in the statistical approach, the τ parameter can be set by the user and has the same meaning as described in 5.3.1. The pseudo algorithm used for the shadow detector in the statistic implementation is as follows:

For each pixel that is initially classified as foreground pixel the following operations are executed to see if the pixel is a shadow pixel candidate:

- repeat for all used components in the background model of the pixel (thresholded by parameter T in equation (19):
  - get the variance, the weight and the mean values for R,G and B.
  - let $c = (R, G, B)$ and $\mu = (\mu_r, \mu_g, \mu_b)$ calculate a ratio of similarity between the current pixel R,G and B values and the means in the Gaussian component :

$$S_{RGB} = \frac{c\mu^T}{\|\mu\|^2} \tag{21}$$

    (see appendix A for a graphical derivation of equation 21)

    When the current pixel is more intensive than it's mean, S_{RGB} will be greater than 1. if the current pixel is darker S_{RGB} will always be between 0 and 1.
  - if S_{RGB} < 1 and S_{RGB} > τ then the current pixel is a candidate shadow pixel. A final check is needed with respect to colour distortion. This is done by first calculating the square distance vector between vector c and $S_{RGB}\mu$:

$$D = \left|S_{RGB} \cdot \mu - c\right|^2 \tag{22}$$

    and then classifying the pixel as shadow pixel if and only if

$$D < m\sigma_K S_{RGB}^2 \tag{23}$$

  - where m is the squared Mahalanobis distance which was described in chapter 5.2. σ_K is the variance of the current Gaussian component. In appendix B, equations (25) and(26) are rewritten for optimized implementation and also graphically is shown why this condition should hold. Update the mask at corresponding location (write value 125) break repeat
  - else the pixel is a pure foreground pixel, continue with the next Gaussian component
- If all Gaussian components are processed and the pixel is still not classified as a shadow pixel, classify the pixel as a pure foreground object. update the mask at corresponding location (write value 255).

5.4. Video Summarization

For both background subtraction implementations, a video summarization option is included. With this option enabled, which can be activated with the `R` or `Rb` parameter (see chapter 5.5), video frames which includes "enough moving objects" are written to a video file on disk. The idea behind this feature, is that frames which do not contain enough movement are not recorded. This results in a video file which only contains movements of interest and excludes all frames where no traffic was present at all. To determine if the current frame needs to be stored, we simply count the number of pixels in the binary mask `M(x,y)` that have a value of 255 (classified as a pure foreground pixel). If this number of pixels exceeds the value `R` (or `Rb`), the current frame is written to the movie file on disk. The choice of R (or `Rb`) is scene dependent. For example if a lot of branches of trees in the scene are moving in the wind (which results in some foreground misclassifications) a higher value for this parameter is desired.

All the frames that are stored by the video summarization system are labelled by a second-precision timestamp (see figure 10) and appended to the output movie file. In this way, when the recorded result is watched later, the time intervals in which no traffic was present can be determined easily. The output file can be written to disk using compression in real-time. The sample files we produced in this research are compressed using the XVID codec which only requires on average 1.5 MB for a minute of video in 640x480. If the video summarization option is enabled by the user, a codec for compression can be selected from a window. The output file will be stored in the same directory as the main executable and the filename for the video has the format: `[yyyy-mm-dd]-hh.mm.ss.avi`

Figure 10 – *Recorded video frame labelled with a timestamp.*

Now suppose one car enters the scene. The video summarization system described above will start recording as soon as already a significant part of the vehicle is visible in the scene (depending on the value given for parameter `R`). Since we are interested in the recording of the *full path* of the vehicle in the scene, we extended the given basic idea by keeping a buffer of history frames present which can be written to the summarization video when this is necessary. In analogy, when the car will leave the scene the basic system will already stop recording when the car is still partly visible. In the extended implementation the recording will stop when the car has completely disappeared from the scene. The flowchart in figure 12 shows how this extension of the basic system is implemented. This extended video summarization can be activated by using the `Rb` parameter instead of the `R` parameter. See section 5.5 for more information.

5.5. How to use the software

To compile the two background subtraction implementations, open the corresponding project files with Visual C++ and make sure that these include paths are set in
`tools > options > directories > include files`

- `<opencv directory>\cv\include`
- `<opencv directory>\cxcore\include`
- `<opencv directory>\otherlibs\cvcam\include`
- `<opencv directory>\otherlibs\highgui`
- `<source code directory>\bloblib`

Also make sure that the following libraries are set in
`tools > options > directories > library files`

- `<opencv directory>\lib`
- `<source code directory>\bloblib\lib`

Finally use the libraries in the project itself by setting the library filenames in 'object/library modules' (`project > settings > link tab`)

- `cvblobslib.lib CvBSLib.lib cv.lib cxcore.lib highgui.lib cvcam.lib`

note: `CvBSLib.lib` is only required for the statistical background implementation.
The binaries themselves can be run using command line parameters. When the program is executed using no parameters, a list of parameters is shown. To display the available cameras attached to the PC and their `cam_number`, type:

- `traffic.exe –cam list`

To do background subtraction with default additional parameters, use one of the following set of required parameters:

- Do background subtraction using a live stream from a webcam:
  `traffic.exe` **`–cam <cam_number>`**
- Do background subtraction on a local video file (avi) on disk:
  `traffic.exe` **`–mov <movie_path>`**
- Do background subtraction on a local video file (avi) on disk and loop the movie:
  `traffic.exe` **`–mov <movie_path> -loop`**

The deterministic background subtraction program also supports some additional parameters that can be set. Place them after the required parameters:

- **`-N <value>`** number of frames to create the initial background model (default value: 5)
- **`-T <value>`** subtraction threshold: I(x,y) - B(x,y) < T (default value: 40)
- **`-I <value>`** initial background model: 0: use mean, 1: use median (see A2 and A3) (default value: 1)
- **`-S <value>`** value for τ , the threshold for shadow darkness as described in chapter 5.3.1 (default value: 0.5)
- **`-Sf <value>`** size of the history buffer used for calculating statistics needed in the shadow detection algorithm.
- **`-O <value>`** output: 0: binary mask only, 1: mask and current frames (default value: 0)
- **`-B <value>`** show bounding boxes around foreground objects, 0: no, 1:yes (default value: 0)
- **`-R/-Rb <value>`** Record those frames where the number of classified pure foreground pixels exceeds <value>. A window will popup where the user can select the codec. Use `–Rb` for pre- and post recording. (default: no recording)
- **`-clean`** show the incoming stream without any operations performed. This can be used to show the original speed (fps) of the stream.

The statistical background subtraction program also supports some additional parameters that can be set. Place them after the required parameters:

- **-T <value>** size of the frame buffer; number of history elements. Alpha = 1/T. (default value: 1000)
- **-C <value>** threshold on the squared Mahalanobis distance to decide if it is well described by the background model or not (default value: 64)
- **-S <value>** value for τ , the threshold for shadow darkness as described in 5.3.2 (default value: 0.5)
- **-O <value>** output: 0: binary mask only, 1: mask and current frames (default value: 0)
- **-B <value>** show bounding boxes around foreground objects, 0: no, 1:yes (default value: 0)
- **-R/-Rb <value>** Record those frames where the number of classified pure foreground pixels exceeds <value>. A window will popup where the user can select the codec. Use `-Rb` for pre- and post recording. (default: no recording)
- **-clean** show the incoming stream without any operations performed. This can be used to show the original speed (fps) of the stream.

The result of the algorithm on each video frame is shown in a window. The program closes automatically as soon as the stream of incoming frames ends or when the user closes the application.

Also, for easier use of the software, a combined GUI (figure 11) for both background subtraction methods has been created in which the parameters can be set in a more intuitive way. Parameters T, C, S of the statistical algorithm and parameters N, T, I, S of the deterministic algorithm can be adjusted in a window that will appear when the "Additional Parameters" button is pressed. The more basic parameters like the choice of the algorithm and the output mode (O) can be accessed directly from the main panel. Also the video summarization component can be activated (parameter R) from this main panel, which enables recording only frames in which enough foreground pixels are present, as was described in section 5.4.

A configuration of parameters can be stored to disk using the "Save Preset" button and reloaded by using the "Load Preset" button. The "Test Camera" button corresponds to the `-clean` parameter which will just show the incoming unprocessed video frames. Enabling the shadow detection option in the GUI corresponds to the setting of parameter `S` to value 1. The "use pre- and postrecording technique" option enables the extended video summarization described in section 5.4 and displayed in figure 12.

6. Evaluation

In this chapter the two implemented background subtraction methods described in chapter 5 are evaluated using recorded movie files. The movies were recoded with a Philips ToUcam Pro webcam. In 6.1 the characteristics of the movies are described, in 6.2 we discuss for each of the two algorithms which parameter settings are optimal, given the types of videos. In section 6.3 a comparison measure is proposed to actually compare the two subtraction methods. Finally in 6.4 both implemented algorithms are compared using a given performance measure

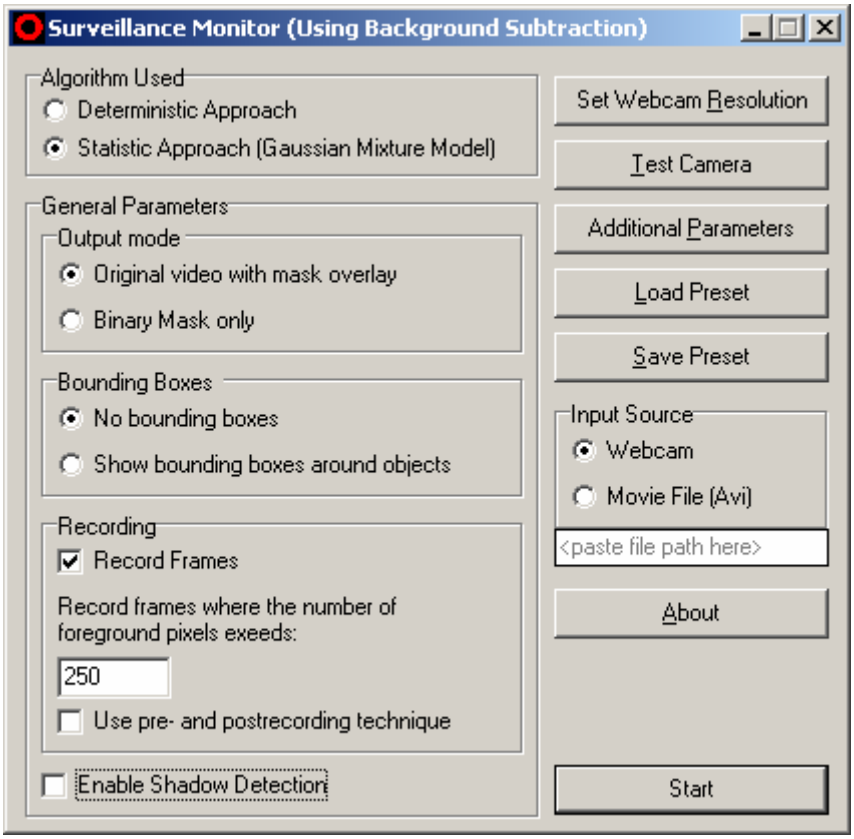


Figure 11 – *GUI for easy control of both background subtraction methods and their parameters*

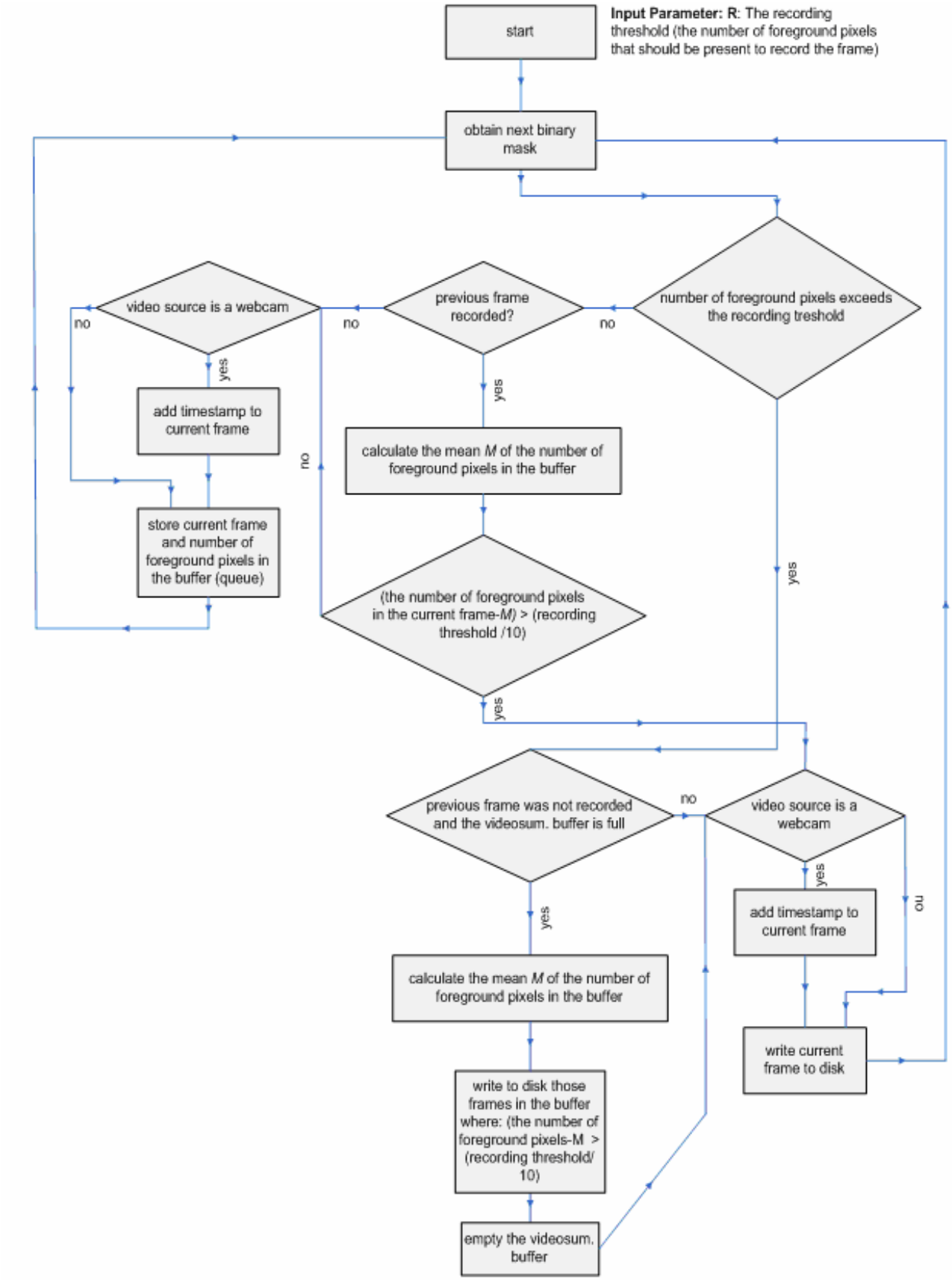

start
Input Parameter: R: The recording threshold (the number of foreground pixels that should be present to record the frame)
obtain next binary mask
number of foreground pixels exceeds the recording treshold
previous frame recorded?
video source is a webcam
no
yes
add timestamp to current frame
store current frame and number of foreground pixels in the buffer (queue)
calculate the mean M of the number of foreground pixels in the buffer
(the number of foreground pixels in the current frame-M) > (recording threshold /10)
previous frame was not recorded and the videosum. buffer is full
video source is a webcam
add timestamp to current frame
calculate the mean M of the number of foreground pixels in the buffer
write current frame to disk
write to disk those frames in the buffer where: (the number of foreground pixels-M > (recording threshold/ 10)
empty the videosum. buffer

Figure 12 – *Flowchart showing the extended video summarizing algorithm*

6.1. Types of videos

As stated in the introduction, this research has its focus on traffic monitoring in an urban setting. Typical interesting traffic points in cities are crossroads. Three videos are recorded looking down on a typical small crossroad in an urban area. In each of the videos the weather conditions are different: sunny, clouded and rainy. The camera that has been used is a Philips ToUcam Pro webcam with a capture resolution of 640x480 and its frame rate was set to 15 fps. Still frames of the videos are displayed in figure 13.

A
Weather conditions: clouded, very windy
Length: 5 minutes (4581 frames)
Traffic lights: no
Colour: yes
Resolution: 640x480
Special: also bikers and pedestrians

B
Weather conditions: sunny
Length: 4.5 minutes (4581 frames)
Traffic lights: no
Colour: yes
Resolution: 640x480
Special: also bikers and pedestrians

C
Weather conditions: rainy
Length: 3.5 minutes (3024 frames)
Traffic lights: no
Colour: yes
Resolution: 640x480
Special: also bikers and pedestrians

Figure 13 *– Three video sequences recorded with a webcam that will be used for testing.*

6.2. Parameter tuning

In this section the optimal values for the parameters discussed in 5.5 will be determined for both implementations given the urban traffic setting.

Deterministic implementation

The essential parameters in this implementation are the number of frames to create the initial background model: N, the subtraction threshold: T, whether to use the mean or the

median to create the initial background model: **I**, the threshold τ for the shadow darkness: **s**, and finally the size of the history buffer used by the shadow detection algorithm: **sf**.

For the best performance of the deterministic implementation it is necessary to have a number of frames of the scene without any traffic present. To capture as much information as possible for creating the initial background model, the number of frames N should be set till that frame number in which foreground objects start appearing in the scene. So when the first object comes in the scene at frame 100, N should be set to 99. Directly related to the value of N is the question whether the mean (I=0) or the median (I=1) should be used to construct the background (see section 5.1). In all the videos it can be seen that we have a lot of moving background objects caused by the wind, like moving tree branches. Therefore, for a number of pixels in the scene, outlier values will occur. So using the median approach always results in better initial performance in this setting. In figure 14 the frame N+1 is shown for both mean and median approaches. On this frame background subtraction, as described in 5.1, is applied and the white pixels correspond to the foreground pixels. Since there is still no traffic visible, these pixels are wrongly classified. From both images it can be seen that the median approach of creating the background model performs better since a lower number of (wrongly classified) foreground pixels was found.

Figure 13 *– Processed frame N+1 from video A using initial background constructed using the mean approach (left) and using the median approach (right) In the left image a number of foreground pixels are clearly visible in the area of the upper left tree*

The next parameter of interest is the subtraction threshold parameter T. This is the value for T in pseudo algorithm (A1). By setting T to a relatively low value the probability of detecting the foreground objects increases but also the number of misclassifications in the background increases when the lightning conditions change just a bit. The value for T is highly scene dependent. For our recorded videos A,B and C this value is set to 40.

The values set for the two parameters S and Sf determine the performance of the shadow detector. The value for S ranges from 0 to 1 and represents τ (see section 5.3.1). If this parameter is set to a relatively low value, a lot of foreground pixels will be classified as shadow pixels. If S is set to 1, the shadow detector is disabled since the

condition "if S_{RGB} < 1 and S_{RGB} > τ" will never be valid for any pixel values . S is also a scene dependent parameter. For the videos A,B and C the values are set to: 0.4. The number of history frames (Sf) is used to construct a mean of pixel values at each location in the scene. This results a more robust way to see if a pixel is indeed for example darker for a certain period of time. In this research we use a default value of 15 shadow history frames.

Statistical implementation
The essential parameters in the statistical implementation are the size of the frame buffer (number of history elements to consider): **T**, the threshold on the squared Mahalanobis distance to decide if it is well described by the background model or not: **c**, and finally, the threshold τ for the shadow darkness: **s**

The value for T determines how fast the weight of a Gaussian component changes when a new sample is processed. A high value for T means that the speed of update slows down. In traffic scenes, where a robust background model is needed, the speed of update cannot be too fast and therefore this parameter is set for all videos to 1000. This means that 0.001 is added (if the new sample matched this component) or subtracted (if there was no match) from the weights of a Gaussian component.

The width of the band (in σ) in which a sample will match a certain component is set by the threshold parameter C. The higher the value of C, the lower number of Gaussian components will be created. As seen before, in traffic situations, multiple background components per pixel might be modelled, for example in those locations where we have moving branches of trees. Therefore the size of C should not be too big but also not too small. For all the videos in this research we use a value of 64 for the C parameter. Since the shadow detection algorithm is basically the same for both implementations, the same values for the S parameter in this statistic setting are used as those in the deterministic setting.

6.3. Comparison measure
The most ideal way to test the performance of the implemented algorithms is to first create binary masks manually in which the foreground pixels are highlighted by the user (the ground truth). These binary masks can then be compared with the binary masks created by the background subtraction algorithms. Videos A, B and C altogether contain thousands of image frames. Creating the ground truths manually is therefore too time consuming.

Using the Video Summarization component as evaluation tool
In this research another evaluation measure is chosen. The video summarization component defined in 5.4 is actually used as an evaluation tool. Only frames that contain enough classified foreground pixels are recorded. If for example a frame was recorded which did not contain a foreground object at all, we can conclude that a lot of misclassifications did occur in that frame using the background subtraction algorithm. The steps executed for evaluating the two implemented algorithms using the video summarization component can be defined as follows:

1. Create a frame-level ground truth for each video: extract the frame numbers manually in which one or more foreground objects (vehicles, bikers, pedestrians, etc.) are visible. (the tables in figure 15)

2. Execute the video summarization algorithm described in figure 12. This results in a new video only containing those frames with enough classified foreground pixels. The original frame numbers are also stored (as an overlay on the frame itself)
3. In this summarized video extract the successive intervals of frame numbers and store them (tables of figure 16)
4. Compare the found intervals with the frame intervals determined in the ground truth: count the number of non-overlapping frames. The lower the number of these non-overlapping frames, the better the performance of the algorithm.

6.4. Comparison results on practical video's

For the three videos A,B and C the evaluation steps described in section 6.3 are executed. The three tables depicted in figure 15 contain the ground-truth results for respectively video A, B and C. (step 1 of the evaluation)

Frame number intervals in which foreground objects are visible	*Type of traffic visible in this interval*	*Interval length (in frames)*
421-463	Car	42
647-690	Car	43
777-864	Car and cyclist	87
1023-1107	Cyclists	84
1464-1579	Cyclist	115
2066-2110	Car	44
2213-2264	Car	51
2441-2541	Car and cyclist	100
2634-2672	Car	38
3563-3890	Pedestrian, cyclists and a car	327
4190-4233	Car	43
4316-4386	Cyclist	70
4485-4513	Motorcycle	28

Figure 15a *– ground truth results for* ***video A****, which has a total of 4581 frames and 1072 frames in which foreground objects were visible*

Frame number intervals in which foreground objects are visible	*Type of traffic visible in this interval*	*Interval length (in frames)*
89-170	Car and cyclist	81
321-386	Cyclist	65
697-813	Cyclists	116
887-921	Car	34
999-1251	Car and cyclists	252
1265-1453	Car and cyclists	188
1473-1503	Car	30
1622-1648	Car	26
2205-2445	Cyclists	240
2538-2572	Car	34
2622-2705	Cyclist	83
2843-3319	Car, cyclist and pedestrian	476
3400-3476	Cyclist	76
3794-3876	Cyclist	82
3895-4117	Pedestrian and a dog	222

Figure 15b *– ground truth results for* ***video B****, which has a total of 4163 frames and 2005 frames in which foreground objects were visible*

Frame number intervals in which foreground objects are visible	*Type of traffic visible in this interval*	*Interval length (in frames)*
190-258	Car	68
615-659	Car	44
797-837	Car	40
1111-1155	Car	44
1363-1409	Car	46
1734-1777	Car	43
2058-2508	Car, pedestrian, dog and cyclist	450

Figure 15c *– ground truth results for* ***video C****, which has a total of 3024 frames and 735 frames in which foreground objects were visible*

For videos A,B and C, the results of the deterministic and statistical background subtraction algorithms, together with the ground truths, are displayed in the tables in figure 16. (results of step 3 of the evaluation) In the table for video A it should be noted that the value for parameter `Rb` was set to a high value (1200) due to relatively a high number of misclassifications caused by the swinging tree branches in the wind. (it will take a number of frames before the implemented Gaussian Mixture model can handle the 'swinging tree branches problem')

Frame number intervals in which foreground objects are visible: ground truth	*Recorded frame intervals using the deterministic approach*		*Recorded frame intervals using the statistical approach*
421-463	280-298	2447-2526	279-479
647-690	310-325	2606-2616	647-903
777-864	339-344	2635-2670	1059-1103
1023-1107	367-382	2901-2920	1467-1576
1464-1579	401-410	3196-3204	2067-2109
2066-2110	416-450	3460-3499	2214-2263
2213-2264	455-462	3639-3845	2442-2538
2441-2541	644-915	4018-4039	2634-2671
2634-2672	1052-1104	4188-4217	3641-3682
3563-3890	1467-1558	4226-4244	3692-3767
4190-4233	2074-2099	4282-4296	3793-3845
4316-4386	2103-2110	4327-4383	4190-4218
4485-4513	2210-2283	4484-4500	4226-4232
	2315-2330	4510-4514	4329-4384
	2362-2413		4485-4499

Figure 16a *– results of the deterministic and statistical background subtraction algorithms applied on* ***video A****, together with the ground truth intervals. Value of parameter Rb = 1200*

Frame number intervals in which foreground objects are visible: ground truth		*Recorded frame intervals using the deterministic approach*		*Recorded frame intervals using the statistical approach*	
89-170	2205-2445	90-110	2236-2431	88-192	2622-2704
321-386	2538-2572	132-169	2542-2579	325-384	2743-2769
697-813	2622-2705	353-385	2622-2700	697-812	2785-3011
887-921	2843-3319	735-812	2839-3007	887-921	3020-3131
999-1251	3400-3476	889-929	3022-3125	999-1249	3139-3162
1265-1453	3794-3876	1008-1256	3400-3434	1271-1450	3195-3304
1473-1503	3895-4117	1292-1445	3458-3477	1473-1502	3400-3490
1622-1648		1473-1498	3797-3875	1621-1647	3795-3875
		1624-1648	3900-4124	2206-2436	3895-4114
				2537-2570	

Figure 16b *– results of the deterministic and statistical background subtraction algorithms applied on* ***video B****, together with the ground truth intervals. Value of parameter Rb = 250*

Frame number intervals in which foreground objects are visible: ground truth	*Recorded frame intervals using the deterministic_approach*		*Recorded frame intervals using the statistical approach*	
190-258	8-140	1111-1155	19-20	797-838
615-659	156	1201	42-46	948
797-837	190-412	1359-1408	51-65	1009
1111-1155	536	1588-1589	107	1107-1155
1363-1409	615-659	1736-1776	156-157	1201
1734-1777	780	1867-1868	191-261	1364-1409
2058-2508	798	2148-2172	319	1453-1454
	799-836	2249-2536	368	1588-1589
	946-948		422-423	1734-1776
			482	1867-1868
			536-537	2123-2176
			615-658	2249-2502
			712-713	2805
			780-781	

Figure 16c *– results of the deterministic and statistical background subtraction algorithms applied on* ***video C****, together with the ground truth intervals. Value of parameter Rb = 250*

As can be seen from the table above, miscalculations do occur using the two implemented subtraction methods. To get a better view of the frame intervals of the ground truths and the frame interval results of both algorithms, a graph is plotted for each video in figure 17. In this graph the frame numbers are located on the X-axes, and corresponding bins mark those frame numbers that were recorded (green and red bars). The yellow bars mark the ground truth intervals: in these intervals, one or more objects were present in the scene.

To obtain a score for both background subtraction algorithms, the number of frames in which miscalculations did occur (E), are counted and compared to the total number of frames (N) of the videos:

$$S = (1 - \frac{E}{N}) \times 100 \qquad (24)$$

In this way S is the percentage of frames that were recorded correctly. For videos A, B and C these scores are calculated for the deterministic and statistical subtraction method. This corresponds with the last evaluation step as was stated in section The results are displayed in figure 17.

	Score S (deterministic approach)	*Score S (statistical approach)*	*total number of frames*
Video A	85.2%	88.4%	4581
Video B	88.6%	94.6%	4163
Video C	83.5%	93.4%	3024

Figure 17 *– performance of both subtraction algorithms on the three test videos.*

From the scores we can see that the statistical approach has a better performance in all cases. No matter the weather conditions, the statistical approach outperforms the deterministic approach. Furthermore we see that wind is the most difficult problem for the subtraction algorithms. In video A, a lot of trees and branches are moving in the

wind. A good example can be seen in figure 19. In the graph of video A, interval [241, 482], it can be seen that a lot of frames are recorded (both algorithms) in which actually no foreground objects are visible. Also from figure 19 it can be seen for video A that the deterministic approach makes more misclassifications caused by the wind (for example in the interval [2652, 3616]. In the scene with the sunny weather condition (video B) it can be seen that both algorithms perform very well. The cast shadows were eliminated successfully by the shadow detector. In the scene with the rainy weather condition (video C) it can be seen that there are a number of small misclassification intervals. These are caused by falling raindrops visible in just a few frames, resulting in big trails in front of the camera. As can be seen from figure 19, the statistical algorithm produces a lot of those short misclassified frame intervals, especially in the interval [0, 636]. The deterministic algorithm has more difficulties with rainy situations. In the interval [0, 636] it can be seen that a lot of unwanted frames have been recorded. In this part of the movie the rain is most intense, so it can be concluded that the statistical approach is more robust in rainy scenes.

It is also interesting to look at the different kinds of misclassifications. A misclassification can be either "the frame that was recorded, but actually should not be recorded since no foreground objects were visible" (E_{rec}) or "the frame was not recorded, but the frame actually does contain a foreground object and therefore should be recorded." ($E_{not\ rec}$) In the application of a surveillance monitor the latter misclassification is more severe than the former since it is better to record some unnecessary frames than not recording frames containing possible interesting objects. The ratios between those two types of misclassifications in the test videos, are displayed in figure 18 below.

Mis-classifications	*Deterministic approach* E_{rec} - $E_{not\ rec}$	*statistical approach* E_{rec} - $E_{not\ rec}$
Video A	0.64 – 0,36 [431-247]	0.53 – 0.47 [283-250]
Video B	0.89 – 0.91 [42-432]	0.55 – 0.45 [123-100]
Video C	0.66 – 0.34 [328-170]	0.25 – 0.75 [51–150]

Figure 18 *– ratio between the two different kinds of misclassifications for each of the test videos*

From figure 18 it can be seen that for video A the misclassification ratio of the deterministic approach is better than in the statistical approach (smaller $E_{not\ rec}$) although the total score S in figure 17 is slightly better when using the statistical approach. For video B it can be seen that the misclassification ratio of the deterministic approach is very bad. Almost all misclassifications were frames that should be recorded, but were not. This is mainly caused by the inability of the deterministic approach to detect objects in shaded areas. Finally figure 18 shows for video C that the ratio of misclassifications in the statistical approach seems disappointing but it should be noted that the number of E_{rec} was very low in this case. Comparing the absolute values between both approaches (150-170) shows a somewhat equal number of misclassifications of this type.

The video summarizations of video's A B and C can be downloaded from the project website [29]. Also the background subtraction result (foreground pixels highlighted) for each frame of the three videos can be accessed from the website. In these video the performance of the shadow detector is also visible. The foreground pixels that are highlighted in green are actually classified shadow pixels. In general we see here better results for the statistical background subtraction algorithm in terms of detecting the *whole* object of interest. In the deterministic approach, holes will often appear in the

foreground objects. Furthermore in the sunny scene (video B) it should be noted that when an object is moving through shadowed areas, the deterministic background subtraction approach is often only able to detect parts of the object. Because of the dark regions in that shadow area, colours in that area tend to be more similar.

As described in section 5.1 and figure 6, trails of unwanted foreground pixels might occur using the deterministic subtraction algorithm. Despite of the solution proposed in the corresponding section, still a number of those unwanted pixels are visible in the videos containing the subtraction results. An extension to the solution is that the entire background model will be refreshed as soon as the number of foreground pixels do not change much in an interval of a certain number of successive frames. In the experiments performed the background model is totally refreshed when during five successive frames the number of classified foreground pixels differs no more then 5 pixels.

As said before, creating a ground-truth for each frame, by marking all pixels belonging to a foreground object, is very time consuming. The results of such an evaluation are very interesting and definitely worth analysing, which results in a pixel level evaluation of both background subtraction algorithms. An evaluation on this level is not included in this research and is proposed as future work.

7. Conclusions

In this research, two different background subtraction techniques for traffic monitoring were implemented and compared. The deterministic subtraction technique uses the current pixel values in a video frame to update the background model. To make sure that moving foreground objects are not registered in the background model, only the pixels classified as background are updated. Classifying is done by considering the distance between pixel values in the background model and the new incoming video frame. The second implemented technique is a statistical approach which models each pixel in the background model by a Gaussian mixture model. Updating the model is done by a number of update equations that update the weight, mean and variance of each Gaussian component in the model. Classifying a new pixel is done by extracting those Gaussian components in the model that have high weights and low variances, which are the indicators of being a background component. Both algorithms are evaluated using an implemented video summarization technique. The recorded videos that were used for the evaluation are 3 scenes of the same crossroad under different weather conditions. From this evaluation it can be concluded that the statistical approach has a better performance in all cases. No matter the weather conditions, the statistical approach outperforms the deterministic approach. The deterministic algorithm has difficulties with rainy scenes and scenes in which a lot of branches of trees are moving in the wind. The statistical approach deals better with this problem since multiple Gaussian components can act as background components. Furthermore the shadow detector works very well in the test scenes, as can be seen in the videos showing the subtraction results. The software developed works in real time using a webcam stream (640x480) or an arbitrary movie file as input. Using the webcam, the system can be used as a smart surveillance system where only frames are recorded when movement, detected by one of the two implemented subtraction algorithms, is present.

8. Future work

This research aimed at the background subtraction part of an urban traffic monitoring system. As displayed in figure 2, a lot of other components are needed for the proposed system. The binary mask that separates the background from the foreground is the result of the subtraction algorithms in this research. This data should be used by a tracking system that is able to track each object of interest in the scene separately. Furthermore it would be nice if the tracker system is able to detect the same vehicle in different scenes. In that way the actual flow of each vehicle in the city can be monitored. Having the tracking results, traffic parameters like speed, density and classification of a vehicle can be extracted and stored. Proposed improvements within the background subtraction algorithm are to prevent queuing vehicles to be counted as background objects if those vehicles are not moving for a period of time. A possible solution is to select regions in the scene where newly detected objects of a certain size should never be incorporated in the background model. Another improvement might be found in the choice of another colour model in the pre-processing step. Transforming the RGB space to normalized RGB or the HSV colour model might result in better subtraction results, especially when illumination changes often occur. Finally the evaluation of the implemented algorithms in this research could be extended by evaluating them on scenes for which manually annotated ground truth frames are available. The results of such an evaluation could be very interesting and is definitely worth analysing, which results in a pixel level evaluation of both background subtraction algorithms.

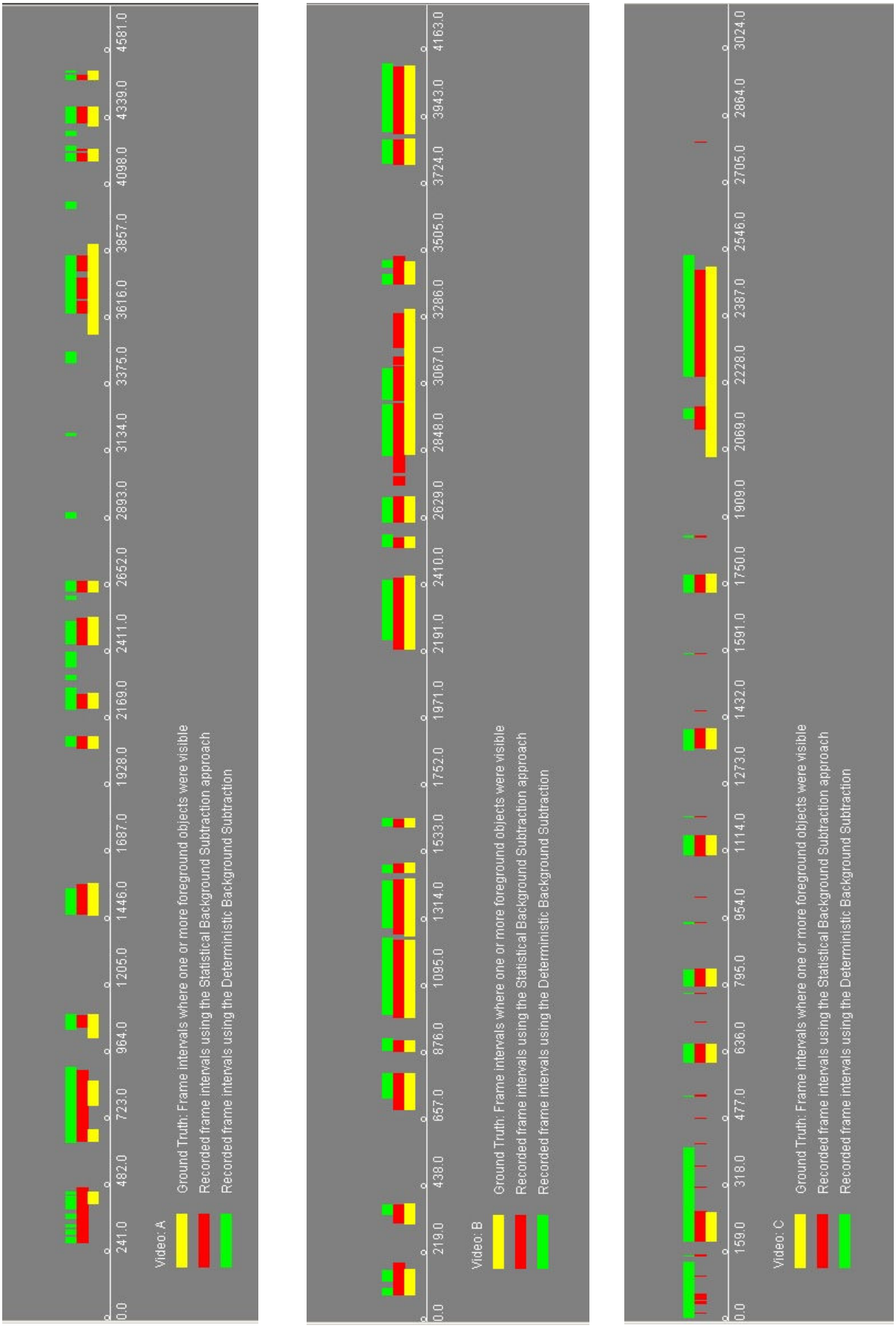


Figure 19 – *graphs displaying recorded frame intervals using the two background subtraction methods. The yellow bars mark the intervals where one or more objects are present in the scene*

9. References

[1] S. Gupte, O. Masoud, R.F.K. Martin, N.P. Papanikolopoulos, "Detection and Classification of Vehicles", IEEE Transactions On Intelligent Transportation Systems, Vol. 3, No. 1, March 2002

[2] R. Cucchiara, M. Piccardi, "Vehicle Detection under Day and Night Illumination", Proceedings of 3 rd International ICSC Symposium on Intelligent Industrial Automation (IIA), 1999

[3] M. Betke, E. Haritaoglu, L.S. Davis, "Multiple Vehicle Detection and Tracking in Hard Real Time", IEEE Intelligent Vehicles Symposium, pp. 351—356, July 1996

[4] D.M. Ha, J.-M Lee, Y.-D. Kim, "Neural-edge-based vehicle detecting and traffic parameter extraction", Image and Vision Computing, Vol. 22, 2004, pp. 899–907.

[5] A.J. Lipton, H.Fujiyoshi, R.S. Patil, "Moving Target Classification and Tracking from Real-time Video", In 4th IEEE Workshop on Applications of Computer Vision, pp. 8–15, 1998

[6] Z. Zhu, G. Xu, Bo Yang, D. Shi, X. Lin, "VISATRAM: a real-time vision system for automatic traffic monitoring", Image and vision computing [0262–8856] Zhu, vol:18 pg:781, 2000

[7] M. Kilger, "A shadow handler in a video-based real-time traffic monitoring system", In Proc. IEEE workshop on Applications of Comp. Vision, 1992, pp.11–18

[8] P.L. Rosin, T. Ellis, "Image difference threshold strategies and shadow detection", In Proc. of the sixth British Machine Vision Conference, 1994

[9] P. Kaew Tra Kul Pong, R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection", In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS 01. Sept 2001.

[10] A. Prati, I. Mikic, c. Grana, M.M. Trivedi, "Shadow Detection Algorithms for Traffic Flow Analysis: a Comparative Study", In Proc. IEEE Intelligent Transportation Systems, 2001

[11] T. Horprasert, D. Harwood, L.S. Davis, "A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection", Proc. IEEE ICCV'99 FRAME-RATE Workshop, Kerkyra, Greece, September 1999.

[12] E. Salvador, A. Cavallaro, T. Ebrahimi, "Shadow Identification and Classification using invariant Color Models", In IEEE Signal Processing Soc. Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 2001), IEEE Press, 2001, pp. 1545–1548

[13] A. Prati, I. Mikic, M.M. Trivedi, R. Cucchiara, "Detecting Moving Shadows: Algorithms and Evaluation", IEEE Transaction On Pattern Analysis and Machine Intelligence, Vol. 25, No. 7, July 2003

[14] A. Elgammal, D. Harwood, L. Davis, "Non-parametric Model for Background Subtraction", In Proc. of the 6th European Conference on Computer Vision-Part II, Pages: 751 - 767, 2000

[15] S-C.S. Cheung, C. Kamath, "Robust Background Subtraction With Foreground Validation For Urban Traffic Video", In Journal: Eurasip Journal on Applied Signal Processing; Journal Volume: 14, 2004

[16] R.T. Collins, A.J. Lipton, T. "A System for Video Surveillance and Monitoring", tech. report CMU-RI-TR-00–12, Robotics Institute, Carnegie Mellon University, May 2000

[17] S-C.S. Cheung, C. Kamath, "Robust Techniques For Background Subtraction In Urban Traffic Video", In Proceedings of the SPIE, Volume 5308, pp. 881–892, 2004

[18] D. Beymer, P. McLauchlan, B. Coifman, J. Malik, "A Real-time Computer Vision System for Measuring Traffic Parameters", In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997.

[19] M.B. van Leeuwen, F.C.A. Groen, "A Platform for Robust Real-time Vehicle Tracking Using Decomposed Templates", Intelligent Autonomous Systems Group, Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands, unpublished, 2001.

[20] T. Gevers, "Color in Image Search Engines", Survey on color for image retrieval from Multimedia Search, ed. M. Lew, Springer Verlag, January, 2001

[21] T. Gevers and A. W. M. Smeulders, "Color-Based Object Recognition", In Proc. of Pattern Recognition, (32): 453–464, 1999.

[22] R. Cucchiara, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams", IEEE Transactions on Pattern Analysis and Machine Intelligence 25, pp. 1337–1342, Oct 2003.

[23] L.M. Fuentes, S.A. Velastin, "From tracking to advanced surveillance", In Proc. 2003 International Conference on Image Processing, Volume 3, Issue , 14–17 Sept. 2003 Page(s): III - 121–4 vol.2

[24] A.W.K. So, K-Y.K. Wong, R.H.Y. Chung, F.Y.L. Chin, "Shadow Detection For Vehicles By Locating The Object-Shadow Boundary", In Proc. IASTED Conference on Signal and Image Processing (SIP 2005), pp.315–319, 2005.

[25] F. Porikli, J. Thornton, "Shadow Flow: A Recursive Method to Learn Moving Cast Shadows", IEEE International Conference on Computer Vision (ICCV), ISSN: 1550–5499, Vol. 1, pp. 891–898, October 2005

[26] C. Stauffer, W.E.L. Grimson, "Adaptive background mixture models for real-time tracking", In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, pp. 246–252, 1999.

[27] P. Kaew Tra Kul Pong, R. Bowden, "A Real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes", In Journal of Image and Vision Computing. Vol 21, Issue 10, pp913–929, Sept 2003

[28] Z. Zivkovic, "Improved Adaptive Gaussian Mixture Model for Background Subtraction", In Int. Conference Pattern Recognition, Vol.2, pages: 28–31, 2004.

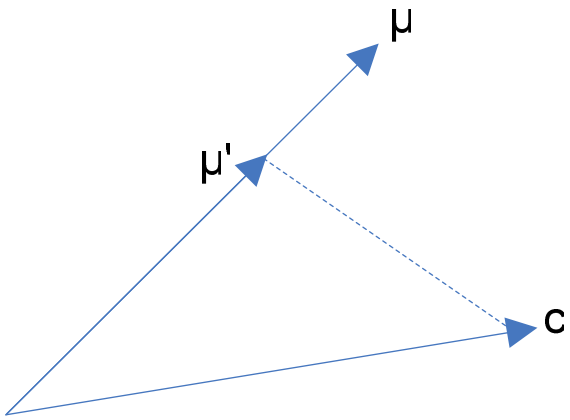
[29] Website for this research including digital version of this paper, sample videos, test results and the binaries of the implementations: http://www.science.uva.nl/~msmids/afstuderen/master

Appendix A. Graphical derivation of the S_{RGB} formula

let $c = (R, G, B)$ and $\mu = (\mu_r, \mu_g, \mu_b)$ the implemented ratio of similarity between the current pixel R,G and B values and their calculated means is:

$$S_{RGB} = \frac{c^T \mu}{\|\mu\|^2} = \frac{R\mu_R + G\mu_G + B\mu_B}{{\mu_R}^2 + {\mu_G}^2 + {\mu_B}^2}$$

The ratio can be graphically shown as being the ratio between $\|\mu\|$ and $\|\mu'\|$



After normalization and using the in-product this ratio becomes:

$$S_{RGB} = \frac{c^T \left(\frac{\mu}{\|\mu\|} \right)}{\|\mu\|} = \frac{c^T \mu}{\|\mu\|^2}$$

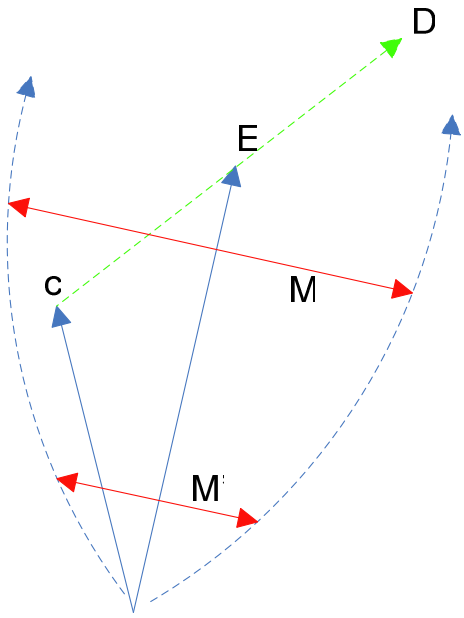
Appendix B. Rewritten formula for colour distortion

let $c = (R, G, B)$ and $E = (\mu_r, \mu_g, \mu_b)$ A check with respect to colour distortion is done by first calculating the square distance vector between vector c and $S_{RGB}E$:

$$D = \left| S_{RGB}E - c \right|^2 = [(S_{RGB}\mu_R) - R)]^2 + [(S_{RGB}\mu_G) - G)]^2 + [(S_{RGB}\mu_B) - B)]^2$$

Again, no square root operation is present in the right term, which optimizes the execution of the program.

Distances D (green) and M = $m\sigma_K S_{RGB}^2$ (red) can be visualized like this:



Note that the position of vector M depends on the value of S_{RGB}. For lower values of S_{RGB} the vector M will be moved down perpendicular to E (for example M')